DREO CR 1999-117

DIINS ADM Development
Phase 1 Report:
System Design Review
Contract: W7714-6-0043/001-SV

SYSTEM DESIGN REVIEW

102-02438/

97-0119-K

DIINS ADM Development

PHASE 1 Report: System Design Review

Contract: W7714-6-0043/001-SV

30 January 1998

Prepared for:
Defence Research Establishment Ottawa
Department Of National Defence
Ottawa, Ontario K1A 0Z4

Prepared by:
PRIOR Data Sciences Ltd.
240 Michael Cowpland Drive
Kanata, Ontario
K2M 1P6

Prepared by: _____
Gary Jackson
Senior Systems Analyst

Reviewed by: _____
Tom Davis
Project Manager

Approved by: _____
Gerry Dawson
Quality Assurance Engineer

PRIOR Data Sciences Ltd.
240 Michael Cowpland Drive
Kanata, Ontario K2M 1P6
Phone (613) 591-7235
Fax: (613) 591-0343

PACKING SLIP

No.: 5726
Date: February 10, 1998

Customer Ref.:	Shipped	FOB	PRIOR Ref.
W7714-6-0043/001/SV	email	Dest.	121129-200035

SOLD TO		SHIP TO	Department of National Defence Defence Research Establishment Ottawa Shirley's Bay, Building T86 3701 Carling Avenue Ottawa, Ontario K1A 0Z4 Attention: Jeff Bird, Scientific Authority Telephone: (613) 990-8906

Item No.	Description	Qty Shipped
1	DIINS Phase I Report (Final), Revision 1.0 dated 30 Jan 98, PRIOR Document No. 97-0119-K	1

Quality Assurance: G. Dawson

c.c.: PRIOR - T. Davis; G. Dawson

REVISION NOTES

VERSION	DATE	DESCRIPTION
0.1	10 Sept 97	First Draft of the Phase 1 Report
0.2	10 Sept 97	Second Draft of Report
0.3	19 Dec 97	Third Draft of Report, for customer review
1.0	30 Jan 98	Formal release of report

TABLE OF CONTENTS

Section Page

1. INTRODUCTION 1-1
   1.1 Scope Of The DIINS ADM Development 1-1
   1.2 DIINS Project Background 1-1
   1.3 Results of DIINS XDM Sea Trials 1-2
   1.4 Acknowledgments 1-2

2. REFERENCE DOCUMENTS 2-1

3. PHASE 1: SYSTEM DESIGN REVIEW 3-1
   3.1 System Design Review Methodology 3-1
   3.2 Kalman Filtering and System Accuracy 3-2
   3.2.1 Introduction 3-2
   3.2.2 DIINS Accuracy Estimate 3-3
   3.2.3 DIINS Filter Configuration 3-3
   3.2.4 Sensor Error Models 3-10
   3.2.5 DIINS ADM Test Philosophy 3-10
   3.3 System Software 3-11
   3.3.1 DIINS XDM Software 3-11
   3.3.2 Current Status Of The XDM Software 3-13
   3.3.3 Enhancements For The DIINS ADM Software 3-14
   3.4 System Components 3-21
   3.4.1 Introduction 3-21
   3.4.2 Processing Component 3-22
   3.4.3 Motorola MVME 177-004 Processor 3-24
   3.4.4 Force SYS68K ISIO-2 Serial Interface 3-24
   3.4.5 Force SYS68K ISIO-1FP Serial Interface Transition Module 3-24
   3.5 Processing and Memory Capacity 3-25
   3.5.1 Processing Requirements 3-25
   3.5.2 Memory Requirements 3-28
   3.6 Cost 3-28
   3.6.1 Cost Comparisons Of ADM Hardware Solutions 3-28
   3.6.2 Alternative Software Development Environment 3-32
   3.6.3 Equipment Spares 3-33
   3.6.4 Cost Observations/Conclusions 3-33
   3.7 Delivery Lead Times 3-35
   3.8 Human-Computer Interface 3-35
   3.8.1 Introduction 3-35
   3.8.2 DIINS Operator Roles 3-36
   3.8.3 Simulation Setup Files 3-37
   3.8.4 Filter Parameter Files 3-39
   3.8.5 HCI Prototype Forms & Screens 3-41

3.9 Sensor Interfaces 3-49
3.9.1 GPS/dGPS Interface 3-49
3.10 System Improvements 3-50

4. DIINS ADM SYSTEM DESIGN SUMMARY 4-1

5. LIST OF ACRONYMS 5-1

LIST OF FIGURES

Figure Page

3-1 DIINS ADM Context Diagram 3-14

C-1 Enhanced DIINS Preliminary Design C-2
C-2 Sperry MK29 1 & 2 (INS) C-6
C-3 Internav LC360 (Loran) C-7
C-4 Trimble TANS PY (GPS) C-8
C-5 Real Sensor Generic C-10
C-6 NMEA Output Interface C-13
C-7 Enhanced Output Interface C-15

D-1 Top Level DIINS HCI Form D-5
D-2 Sensor Selection Form (from installed sensors) D-6
D-3 DIINS Maintenance Mode Menu Selections D-7
D-4 Sensor Installation Form D-8
D-5 Sensor Level Arm Setup Form D-9
D-6 Sensor Communication Channel Setup Form D-10
D-7 FDI Parameter Setup Form D-11
D-8 Data Logging Setup Form D-12
D-9 Failure Injection Form D-13
D-10 System Status Form D-14

LIST OF TABLES

Table Page

3.2.2-1 DIINS ADM Performance Estimate 3-3

3.3.3.6-1 XDM Logged Data 3-20

3.4.2-1 MIPS Relationship to CPU Type and Clock Speed 3-23

3.5.1.2-1 680x0 v. PowerPC Processor Board Comparison 3-26

3.6-1 Option 1: 68040 Based DIINS ADM Solution (same as XDM solution) 3-29
3.6-2 Alternative DIINS ADM Processor Board Solutions 3-30
3.6-3 Possible GreenHills/WindRiver Systems VxWorks Based Solution For The ADM 3-33

3.8-1 Simulation Setup File Parameter Summary 3-38
3.8-2 FilterInit OPVAL File Parameters - Role Applicability 3-40
3.8-3 Filter Configuration File Parameter - Role Applicability 3-41

LIST OF APPENDICES

Appendix Page

A REQUIREMENTS FOR DIINS A-1
B DIINS CONFIGURATION FILES B-1
C DESCRIPTION OF ENHANCED DIINS GENERIC ARCHITECTURE C-1
D HCI PROTOTYPE FORMS D-1
E DATA LOGGING REQUIREMENTS E-1
F DIINS XDM SOFTWARE OPERATION F-1
G DIINS ADM SOFTWARE OPERATION G-1
H NEW SHINNADS TO DIINS INTERFACE REQUIREMENTS H-1
I DIINS ADM SENSOR CONFIGURATION DATA ORGANIZATION I-1

1. INTRODUCTION

1.1 Scope Of The DIINS ADM Development

This report records the results of the studies conducted to review the system requirements and design during PHASE 1 of the development of the Dual Inertial Integrated Navigation System (DIINS) Advanced Development Model (ADM). The studies reported in this document were conducted in accordance with the Statement Of Work given in Reference[1] and the PRIOR proposal, given in Reference[2]. The DIINS software has been developed in the Ada programming language for the DIINS XDM testing and trials. DIINS contains an implementation of multiple parallel Kalman filters that optimally integrate the two MK-29 Inertial Navigation Systems, a GPS receiver and, optionally, other navigation sensors such as Loran C, Omega, and Doppler Speed Log. For details of the DIINS Kalman filter algorithms and the failure detection algorithms see References [5] and [6]. The position, velocity and heading solutions produced by DIINS are used to drive an electronic chart system known as the Shipboard Integrated Navigation and Display System (SHINNADS).

The requirement addressed by the studies documented in this report was to investigate the system requirements and design for the ADM based on the resulting system design and experience gained during the XDM development. The results of the analysis reported in this document will be used to drive the developments in the subsequent phases of the ADM development, namely:

- PHASE 2: Physical Configuration Definition
- PHASE 3: Preliminary Design
- PHASE 4: Detailed Design and Prototype
- PHASE 5: Fabrication

1.2 DIINS Project Background

The following project background information is reproduced from Reference [1].

A need for an integrated navigation system for the Canadian Navy's major warships has existed for some time. Present navigation instruments on the ships operate independently and significant operator effort is required to monitor the navigation suite and determine the best estimate of the ship's true position. The Defence Research Establishment Ottawa (DREO) has conducted an extensive research and development program into the development of a DIINS system that integrates the data from the various on-board navigation sensors and automatically computes the best estimate of the ship's true position, speed, heading and attitude. At the same time the system monitors the health of the various sensors to determine even very subtle sensor abnormalities, and can reconfigure the sensor suite so that the navigation solution is not corrupted by faulty sensor data.

The DIINS developed at DREO is considered an experimental development model (XDM) but it has undergone extensive laboratory testing, dedicated sea trials aboard a research vessel, and an operational evaluation on one of the Navy's Halifax class frigates, HMCS Calgary. The current DIINS XDM software consists of approximately 30,000 lines of Ada code, and the real-time hardware consists of 3 VME bus computer cards and associated components. A separate effort has led to the development of SHINNADS - the Shipboard Integrated Navigation and Display System. SHINNADS - which is being built by Offshore Systems Ltd. of Vancouver B.C. - is an ECDIS (Electronic Chart and Display Information System) that has been specially modified to operate in the navy environment. Now, an Advanced Development Model (ADM) of DIINS is to be designed and built to be inserted into the SHINNADS console in order to provide highly accurate and reliable navigation data to drive the electronic charts of SHINNADS.

1.3 Results of DIINS XDM Sea Trials

For a discussion of the results of the DIINS XDM sea trials see Reference[7].

1.4 Acknowledgments

PRIOR would like to acknowledge the support given during the system design analysis by the DREO DIINS staff, in particular Chrisopher McMillan, Jeff Bird, Lum Hwang and Dale Arden.

PRIOR would also like to acknowledge the contributions made to this report by Harvey Russel of Offshore Systems and Bruno Scherzinger of Applanix.

2. REFERENCE DOCUMENTS

[1]	Request For Proposal Dual Intertial Integ Navigation	Solicitation No. W7714-6-0043 A OBS Ref No. PW-SSSV-041-346
[2]	Management & Technical Proposal Dual Inertial Integrated Navigation System	PRIOR Doc. No. K21049
[3]	Techniques for Soft-Failure Detection in a Multi-sensor Integrated System	J. Christopher McMillan, Jeffrey. S Bird Dale A.G.Arden, Navigation - Journal Of The Institute Of Navigation, Vol. 40, No. 3, Fall 1993
[4]	DIINS Software Enhancement Study: Final Report	Contract W8462-2-FJ0A/02-ER Task: SESS-010, 29 March 1996 PRIOR Doc. 95-0171
[5]	Kalman Filter Design Of Dual Inertial Integrated Navigation System (U)	D.A.G. Arden Dale Arden Consulting, Ottawa Report No. DAC-02-97 August 1997
[6]	DIINS Failure Detection Isolation And Reconfiguration Procedures (U)	D.A.G. Arden Dale Arden Consulting, Ottawa Report No. DAC-03-97 August 1997
[7]	Analysis Of Real Time Results From DIINS 1995 Sea Trials (U)	D.A.G. Arden Dale Arden Consulting, Ottawa Report No. DAC-01-97 August 1997
[8]	NMEA 0183 Standard for Interfacing Marine Electronic Devices	National Marine Electronics Association Version 2.00, January 1, 1992.
[9]	Air Standardization Coordinating Committee: "The Specification for Evaluation Of The Accuracy Of Hybrid Navigation Systems"	ASCC AIR STD 53/16A, 6 April 1988

3. PHASE 1: SYSTEM DESIGN REVIEW

3.1 System Design Review Methodology

In order to conduct the system design review for the DIINS ADM, the review methodology outlined in the following sections was employed:

- Requirements Review - A review of all of the requirements defined in the DIINS RFP, Ref [1], was carried out, and a consolidated set of system requirements for the DIINS ADM was generated, see Appendix A. This requirements list was reviewed with the staff of DREO, and modifications were made to the requirements as appropriate.

- OSL conducted a review of the DIINS XDM SHINNADS display, and produced prototype screen definitions for the DIINS ADM. These screen layouts were based in part on recommendations for improvements made by the DREO DIINS staff, based on their sea trials experience. See Section 3.8.5 for details of the proposed new screen layouts.

- Applanix reviewed various aspects of the DIINS XDM/ADM, including:

    - Filter Configuration Scheme
    - dGPS Interface
    - Sensor Interface Software
    - System Accuracy
    - Data Logging

The results of the Applanix analysis has been included in various sections in the body of this report.

- DIINS Software Architecture - The results of the original DIINS Software Enhancement Study (Ref[4], in particular Section 7 of the Final Report), were reviewed in the area of the definition of the software architecture. This review was made in light of the DIINS ADM requirements recorded in Appendix A. Changes were made to the architecture as a result of the analysis conducted, and the resulting proposed DIINS ADM architecture is recorded in Appendix C.

- User Interface Prototype - In order to understand the requirements and design in more detail, and to get agreement on the look-and-feel of the new Graphical User Interface (GUI) for the ADM, a prototype of the user interface was generated using Visual Basic. This prototype went through several iterations. Part of this GUI prototyping work was aimed at identifying the parameters that were needed to be made available to the user via the GUI in order to support the initialization and maintenance of the ADM. The results of this prototyping activity are presented in Appendix D.

- Data Logging - A review was made of the DIINS XDM data logging capabilities and of the requirements for data logging in the ADM. The volume, content and format of the data logged in the XDM sea trials was reviewed with the DREO DIINS staff. Estimates were made of the volume of data required to be logged for the ADM and its content was defined, see Appendix E.

- DIINS XDM Software Operation - In order to understand, and agree with DREO, the current operation of the DIINS XDM some Sequence Diagrams of the interactions between the software objects were created in a model using the Rational/Rose Modeller. The results of this modelling are included in Appendix F.

- DIINS ADM Software Operation - In order to define the manner in which the DIINS ADM will behave, and in particular how initialization of DIINS takes place via the user interface, some Sequence Diagrams were generated using Rational/Rose, showing the behavior between the DIINS ADM software objects. See Appendix G for the details of the diagrams produced.

- SHINNADS to DIINS Interface Requirements - Using the definition of the SHINNADS to DIINS XDM interface produced by OSL, which is based on the NMEA interface standard (Ref[8]), the details of the SHINNADS to DIINS ADM interface were defined, see Appendix H. Appendix H defines the NMEA sentence structure that is required to implement the ADM behavior defined in Appendix G using the GUI interface defined in Appendix D.

- Configuration Files - In order to assess the changes required for the ADM in the area of configuration files and the setting of parameters, the results of the DIINS Software Enhancement Study (Ref[4]) were reviewed. The content of the existing XDM configuration files was reviewed (see Appendix B) and discussions were held with DREO staff. As a result of this analysis the approach to handling the configuration information for the ADM was formulated, and is presented in Appendix I.

3.2 Kalman Filtering and System Accuracy

3.2.1 Introduction

DIINS implements multiple Kalman filters called "unifilters" that integrate data from one Inertial Navigation System (INS) and a subset of the available aiding sensors such as the Tasman Global Positioning System (GPS), SHINNADS Differential GPS (dGPS), Loran C, and Omega. One filter blends data from all the navigation sensors. The Fault Detection, Isolation, and Reconfiguration (FDIR) algorithm then selects the filter which provides the best estimate of true position, velocity, and three-axis orientation.

The DIINS filters estimate the INS navigation errors and inertial sensor errors. The performance of DIINS will depend on the correct modelling of the INS position, velocity, platform tilt,

accelerometer, and gyro errors. Another contribution to good performance is accurate alignment of the INS and aiding sensor data. The DIINS XDM demonstrated good performance during its sea trials; hence, the Kalman filter design and navigation data time alignment issues appeared to have been addressed.

As part of the PHASE 1 system design review activities, Applanix have reviewed the DIINS Kalman filter design. The review included an assessment of predicted performance and of recommended Kalman filter design improvements. The following sections indicate in detail the results of the analysis conducted.

3.2.2 DIINS Accuracy Estimate

The accuracy estimates indicated in this section are based on the 1995 DIINS XDM sea trials results as reported in Ref [7], and on Applanix's analysis of the achievable performance. The sea trials report Ref [7] presents a compilation of test procedures and observed DIINS XDM characteristics.

Table 3.2.2-1 presents an estimate of the performance achievable for the DIINS ADM based on the analysis conducted to date. This specification will be updated as experimental data becomes available. It can, however, be used to establish performance goals for the ADM development.

Table 3.2.2-1 DIINS ADM Performance Estimate

Navigation Quantity	Units	With C/A GPS or Loran C	With P/Y-code GPS	With dGPS
North, East position error	meters	100	10	1 - 5
North, East velocity error	m/s	0.5	0.1	0.1

3.2.3 DIINS Filter Configuration

3.2.3.1 Introduction

This section presents the results of the initial filter configuration options that have been examined during the first phase of the DIINS ADM development. The filter configuration used during the XDM development which was assumed to be applicable to the ADM was a 15 filter configuration, which would be fed with INS, Loran, GPS and dGPS data. In practice this filter scheme is very computationally intensive, requiring significant processing resources to be used. In

the XDM filter tests the filter update rate was reduced to stay within the processing capability of the 68040 processor used in the XDM. As part of the DIINS ADM development, alternative configuration schemes and update rates will be examined, in order to determine the scheme(s) that come close to meeting the performance of the 15 filter configuration, yet are implementable within the DIINS ADM development budget and schedule constraints.

This section is concerned with the analysis of the DIINS filter configuration when the following inertial and aiding data are available for processing:

- INS1 and INS2 (MK29, MK49 or WSN5)
- P/Y-code GPS
- Differential GPS
- Loran C

The key issue being addressed here is the requirement for 15 concurrent filters to implement the default DIINS architecture for the above sensors. The default DIINS architecture for this discussion is the DIINS architecture that implements the complete permutations of filters for all inertial and aiding sensors as described in Ref [6]. The DIINS default configuration assumes that all aiding sensors contribute statistically independent data to their respective filters, thereby allowing each filter to compute a statistically independent estimated error state. From these, DIINS is able to conduct statistically independent χ^2 tests and thereby detect and isolate a fault in any one INS or aiding sensor.

In particular, the default DIINS configuration assumes that a P/Y GPS receiver and a differential GPS receiver contribute statistically independent information and have mutually observable error characteristics. From a strictly theoretical perspective, this assumption is true. From a practical perspective, the assumption can become uncertain, depending on factors such as the availability and quality of differential corrections.

The reason why this requirement is being examined in detail is that the single board computer (SBC) required to implement 15 filters having a 1.28-second update rate must have a factor of 3 to 4 improvement in performance over the 68040-based MVME 167 used in the XDM. It has been determined that this performance may only be able to be achieved by migrating to a new SBC, such as a PowerPC-based MVME 2600. However, this would require a change to a new software development environment (SDE) from that used on the XDM, which would significantly increase project costs and schedule risk, hence should be avoided if possible. It has been determined that only a factor of 2-to-3 improvement in performance is likely to be obtained over that of the 68040 configuration by using the 68060-based MVME 177 SBC and the XDM SDE. Hence, using the 68060 processor for the ADM, it may not be possible to run the 15 filter configuration, unless it is run slower than the planned 1/1.28 Hz filter update rate.

3.2.3.2 DIINS Design Philosophy

An important consideration in the review of ADM design alternatives is the DIINS design philosophy. The DIINS design seeks to avoid the transient and settling time of a restarted filter by running parallel filters in steady-state, and selecting the best uncorrupted filter to generate the optimal integrated navigation solution. Hence any design recommendations for a DIINS ADM filter configuration must respect the DIINS design philosophy of maintaining steady-state performance during FDIR actions.

The DIINS design approach ensures that the best available navigation solution is available to the ship from the available navigation sensors. The only time that filter transients are normally induced is during power-on initialization of the DIINS.

If a single DIINS 7 filter configuration is considered, then as a consequence of the GPS failure, the filter would have to be reconfigured to exclude the failed sensor and then possibly be restarted. A restarted Kalman filter will exhibit a transient, followed by convergence to steady-state operation and full accuracy for the filter configuration that can last beyond one hour, depending on the available aiding data. For example, a Kalman filter that excludes the GPS receiver and relies on the next most accurate aiding sensor (in the case of DIINS, the Loran C receiver) can require of the order of 30 minutes or more to converge to a steady state on a dynamically benign platform such as a ship.

3.2.3.3 DIINS ADM Filter Configurations

3.2.3.3.1 Seven Filter Scheme

During the development of the DIINS XDM, the majority of the development work has been performed using a 7 Kalman filter configuration, where the inputs to the filters were as follows:

Filter	1:	INS1 & INS2
	2:	INS1, GPS1, LORAN
	3:	INS2, GPS1, LORAN
	4:	INS1, GPS1
	5:	INS2, GPS1
	6:	INS1, LORAN
	7:	INS2, LORAN

3.2.3.3.2 Fifteen Filter Scheme

In addition, tests have been conducted with the XDM to assess the behavior of the DIINS algorithms when two GPS sensor inputs (from two separate GPS receivers, e.g. one GPS and one dGPS) are processed simultaneously in a 15 filter configuration, as follows:

Filter	1:	INS1 & INS2

2: INS1, GPS1, GPS2, LORAN
3: INS2, GPS1, GPS2, LORAN
4: INS1, GPS1, GPS2
5: INS2, GPS1, GPS2
6: INS1, GPS1, LORAN
7: INS2, GPS1, LORAN
8: INS1, GPS2, LORAN
9: INS2, GPS2, LORAN
10: INS1, GPS1
11: INS2, GPS1
12: INS1, GPS2
13: INS2, GPS2
14: INS1, LORAN
15: INS2, LORAN

The problem with the 15 filter configuration, which takes in both the GPS and dGPS inputs, is that as the number of sensors in combination increases the number of states in the filters increases, resulting in an increased processing requirement. If only INS1, INS2, Loran and GPS are used, then a seven filter configuration can be used, and the maximum number of sensors taken account of in a single filter is three (e.g. INS1, Loran and GPS). When an additional sensor input, e.g. dGPS, is added to the filter bank, then the number of the sensor combinations grows to 15, and the maximum number of sensors in a filter increases to four (e.g. INS1, Loran, GPS and dGPS), resulting in an increase in the number of states associated with the filters, hence an increased computational requirement.

3.2.3.3.3 Eleven Filter Scheme

In order to reduce the computational requirement from the single 15 filter scheme, one option is to use two 7 filter banks. One filter bank would contain the inputs from INS1, INS2, Loran and GPS1, while the other filter bank would contain inputs from INS1, INS2, Loran and GPS2. In practice, three filters in each of these banks would be common, while there would be four different filters in each bank. The filter groups would be as follows:

Common Filters To Filter Banks A and B:
Filter 1: INS1 and INS2
2: INS1 and Loran
3: INS2 and Loran

Bank A Additional Filters:
Filter 4a: INS1 and GPS1
5a: INS2 and GPS1
6a: INS1, Loran, GPS1
7a: INS2, Loran, GPS1

Bank B Additional Filters:

Filter 4b: INS1 and GPS2
5b: INS2 and GPS2
6b: INS1, Loran,GPS2
7b: INS2, Loran, GPS2

The two bank filter scheme (Bank A + Bank B + Common, 11 filters in all) defined above provides for one GPS sensor, the most accurate, to be used to provide the Navigation solution, while the other bank represents hot redundancy to be used in the event of failure of the GPS which is being used for the navigation solution. As all of the filters are running continuously, there is a smooth transition from one filter to another in the event of a failure of one of the GPS sensors, which is also a characteristic of the original 15 filter scheme. The main difference between the 15 filter scheme and the above 11 filter scheme is that GPS1 and GPS2 never occur in the same filter, so one GPS is never used to detect a failure in the other GPS. This is the same situation that arises in the 15 filter scheme when one of the GPS sensors becomes unavailable, and hence the 15 filter scheme degrades to a 7 filter scheme using a single GPS, and where GPS failure detection is achieved by comparison with the other sensors, e.g. the Loran.

3.2.3.3.4 Seven Filters, With Multiplexed GPS Inputs

The filter configuration used could be reduced to just running 7 filters, using a single GPS input (the most accurate) at a time. In this case, if the GPS fails the ADM software would be required to detect this fact and switch to using the other GPS sensor in the original 7 filter configuration. In this case the filter states would have to be fully or partially reset, which violates the DIINS philosophy of avoiding transients by running hot redundant filter configurations.

3.2.3.4 Proposed DIINS Development Approach

During the DIINS ADM development, experiments will take place with the 11 and 15 filter (at a reduced update rate) configurations in order to determine and compare their performance and behavior. This will enable the selection of the best filter scheme for the ADM implementation. Both 68040 and the 68060 boards will be available as GFE during the ADM development to enable experiments in processor and filter configurations to be conducted, in order to select the appropriate ADM software and hardware configuration.

Due to the flexibility of the current DIINS software, whereby the features of the filter configuration and failure detection are defined via parameters rather than code changes, filter configuration changes should be relatively easy. The ADM processor should be able to be configured to perform in either an 11 filter configuration, as described previously, or a 15 filter

configuration via parameter changes only. With the 11 filter scheme, failure detection and reconfiguration will be basically the same as in the case of the 15 filter scheme. As the performance of the various sensors degrades, or the sensors fail, the reconfiguration algorithms will continuously select the "best" filter from the remaining filters that process information only from non-degraded and non-failed sensors.

3.2.3.5 GPS Implications on DIINS Kalman Filter Design

The approach to be adopted in DIINS for the GPS inputs is that a single filter design will be used to handle the three types of GPS input. The proposed approach requires that a single set of Kalman filter error states and process noise models be used to model the errors in dGPS data, P/Y GPS data and C/A GPS data. The models will have different parameters; however, their basic structure (number of states, process noise injection) can be the same. The Kalman filter will need to be completely or partially restarted with the new set of parameters when the GPS sensor data input to a particular filter bank changes.

The dominant source of error in C/A GPS and dGPS data is Selective Availability (SA). This is the purposeful degradation of the GPS signal by the U.S. Department of Defence (US DoD) so that C/A GPS exhibits an accuracy desired by the US DoD. The mechanism for degradation are epsilon-SA, the degradation of the satellite clock drift parameter transmitted by a satellite, and delta-SA, the degradation of the ephemeris parameters transmitted by the satellite.

In C/A GPS data, the full SA errors are manifest. In dGPS data, the SA errors are manifest due to latency in application of the differential corrections. This can range from 0 to 10 seconds, depending on the correction type (i.e. RTCM-104 Types 1 or 9) and the correction transmission rate. The dGPS errors include position error growth proportional to the SA acceleration error multiplied by the square of the effective correction latency. The effective correction latency as applies to the GPS navigation solution (geographic position coordinates and North-East-Down velocities) must be computed from the actual latencies in the corrections of the pseudo-ranges, taking prediction of corrections via correction rates into account. If for example Type 1 corrections are used, then the receiver obtains all pseudo-range in a single Type 1 message. Hence the effective and actual correction latencies are the same. If Type 9 corrections are used, then the receiver obtains three pseudo-range corrections per Type 9 message. The effective correction latency will be larger than the actual latency of any one message. It may, however, be lower than the effective Type 1 correction latency because the Type 9 message frequency is higher than the Type 1 message frequency.

This suggests that the same Kalman filter states can be used to model part or all SA errors present in dGPS or C/A GPS data. To model C/A GPS errors, the full SA error states are included in the INS-GPS position measurement model. To model dGPS errors, the SA error states are scaled as a function of effective correction latency in the INS-GPS position measurement model. If differential corrections drop out, then the SA error states scaling changes to 1. In this manner,

the Kalman filter correctly models the dominant GPS position errors with and without differential corrections.

If these same states are used to model P/Y GPS correlated position errors, then the GPS error model's state parameters must change on the occurrence of P/Y-C/A GPS mode transitions. Assuming that these states implement 1st-order Gauss-Markov models, the parameters for each state comprise the initial estimation uncertainty variance σ_x^2, the correlation time τ and the process noise spectral density q_x. Note that these parameters are inter-dependent, as follows, to obtain a stationary model:

$$q_x = 2\,\sigma_x^2/\tau$$

The measurement model parameters will likely not change. These mode transitions are likely to occur very infrequently, usually when the GPS receiver crypto-keys are entered or cleared. When they occur, the GPS position error states must be re-initialized. In this case, DIINS can restart the Kalman filter with the revised error model parameters. Alternatively the DIINS can restart only the GPS error states. These restart capabilities will require new algorithms to be developed and tested for the ADM. For each reset state, this amounts to clearing the off-diagonal rows and columns associated with the reset state, assigning the initial estimation uncertainty variance to the diagonal element of the covariance matrix associated with the reset state, and assigning the new correlation time in the system dynamics model matrix. In either case, the filter will exhibit a transient as it re-establishes cross-correlations with other error models and eventually settles to steady-state performance.

3.2.3.6 Summary of the Proposed DIINS Configuration

To summarize, the following DIINS configuration is proposed:

1. The DIINS filter algorithms will be executed on a single processor, probably a Motorola 68060.

2. The Kalman filters on the processor will be organized into an 11 or 15 filter configuration, as described previously, depending on the results of the development tests as regards DIINS overall performance.

3. The DIINS ADM filters that process GPS data will implement 1st order Gauss-Markov states that model SA position error. The measurement model will use a scaled coupling of the states into the measurement model, where the scale factor is a function of the effective differential correction latency. The details of this implementation will be determined during the detailed design phase of the project.

4. The DIINS ADM will restart its filters that process GPS data if it detects a transition from P/Y GPS status to C/A GPS status or vice versa, or from P/Y GPS status to dGPS status or

vice versa. A partial restart implies a restart of the GPS position error states only. The filter with the most accurate GPS filtered data will be selected during and following the transition of one of the GPS sensors when two GPS sensors are available.

3.2.4 Sensor Error Models

During the detailed analysis phase of the DIINS ADM project, Applanix will conduct a review of the following existing DIINS error models:

- Sperry MK-29 SINS Error Model
- Trimble GPS Receiver Model
- Loran C Measurement Model

In addition, Applanix will also generate the sensor error model to be used in the DIINS ADM for the Sperry MK-49 SIINS.

3.2.5 DIINS ADM Test Philosophy

This section discusses the DIINS performance assessment done to date, and the planned method for assessing the DIINS ADM performance via test.

During the Phase 1 studies, Applanix reviewed the DIINS XDM sea trials report Ref[7]. The report identified a dGPS receiver as the source of a reference position solution when the trial ship, the Endeavour, was within a maximum baseline radius of the reference receiver at CFB Esquimalt. The report did not produce a summary assessment of DIINS performance. It did provide "pieces" of performance information, from which was constructed the performance specification table presented in Section 3.2.2.

It is planned to define a DIINS ADM test program that uses the following document for the specification of the method of evaluation:

- Air Standardization Coordinating Committee: "The specification for evaluation of the accuracy of hybrid navigation systems", ASCC AIR STD 53/16A 6 April 1988, or its successor.

This document specifies the method for computing the 50% probable position error radius (R50) and the 95% probable position error radius (R95) of a hybrid navigation system. It requires a reference position solution. It specifies the accumulation of a statistical pool of six or more navigation tests, from which the test system error statistics are computed. The document is focussed on aircraft navigation systems, for which each test is a separate flight. Applanix have previous experience in computing the error statistics of navigation systems using this

specification. The following will need to be done as part of a definitive test plan to be prepared during the ADM development work:

1. Modify the procedure for marine applications. It is possible that the DIINS ADM will be tested during a single sea trial during which the required tests for the statistical pools are specified. An example of a test procedure might be the requirement to power down and restart all DIINS navigation sensors between tests so as to achieve turn-on-to-turn-on error statistics.

2. Identify a reference system. This can be a dGPS system. However, its accuracy would be comparable with an operational DIINS navigation sensor. Another candidate is a precision carrier phase GPS solution computed in post-processing. In either case, the range from the land-based reference receiver limits the sea trial operating area.

3.3 System Software

3.3.1 DIINS XDM Software

PRIOR performed the DIINS Software Enhancement Study over a period of approximately one year in the 1995/1996 time frame. Various aspects of the software were reviewed, but in particular the following key areas were investigated:

- Reliability;
- Portability and Supportability/Maintainability;
- Flexibility of the software (parameters and modules); and
- Interfaces.

The first point to be borne in mind with the existing DIINS XDM software is that it has been developed via an evolutionary process within a Research and Development environment, and as such the condition of the DIINS code at the time of the study reflected this evolution, with such tell-tale signs as numerous debug statements left in or commented out, and some large modules remaining that will be more difficult to maintain. In addition, although DIINS is written in Ada 83, it does not take advantage (except in isolated cases) of some of the more attractive features (from a software engineering viewpoint) defined in the Ada language, such as Generics, Abstract Data Types, Private Types and Renaming Declarations. The XDM software architecture is basically a functional design, rather than being object-oriented.

For an understanding of the depth of analysis performed by PRIOR during the DIINS XDM study, see the DIINS study report produced by PRIOR, Reference [4]. The following summarizes some of the issues identified during the study of the DIINS XDM software.

During the DIINS Software Enhancement Study the XDM software was analyzed in detail by a number of software engineering tools to extract various parameters about the software. These

tools were REFINE/Ada, Ada-ASSURED, AdaMAT, Cadre Teamwork and PC-Metric. AdaMAT is a metrics analysis tool, and generates copious metrics on all aspects of the software. In addition, it estimates the Reliability, Maintainability and Portability of the software, based on the various assessment criteria built into the tool. As a result of this analysis, AdaMAT assessed the DIINS XDM software to have scores of 0.51, 0.48 and 0.79 respectively in these assessed areas, with an overall score, considering all criteria, of 0.71. The highest (best) score achievable is 1.0. See Sections 5.4 in the DIINS study report for full details of the analysis conducted by PRIOR using AdaMAT.

The conclusion from the AdaMAT analysis was that the software (according to the AdaMAT criteria) was reasonably portable, but did not score very well in terms of Reliability and Maintainability. In its Reliability assessment, AdaMAT uses criteria on both the Anomaly Management built into the code and on the Simplicity of the code. In the area of Anomaly Management it assesses code features such as default initializations, the use of constrained types, suppression pragma's used and the exception handling built in. In the area of Simplicity, AdaMAT considers issues such as program nesting level, the use of goto's and cyclomatic complexity. When modifications are made to the DIINS XDM software by PRIOR to produce the ADM software, strong adherence to reliable programming practices will be employed in all of the modifications carried out. In addition, where unreliable code is detected, and where practical within the project cost and schedule constraints, modifications will be made to the software to enhance its Reliability.

Of particular interest for the DIINS ADM work required on the software is the low maintainability score, as this indicates that the modification of DIINS will likely be hampered by the lack of maintainability built into the software. Key areas where maintainability issues were identified in the program were the extensive use of Ada "Use" clauses (22 "Use" clauses were in effect in one module examined) and the extensive use of global data, i.e. data not passed to procedures using parameters. When there is such a pool of global variables, it makes it very difficult to modify the software without the risk of creating side affects via the coupling introduced by the global variables. With a large number of global variables the extent of the information that has to be comprehended by the software developer/maintainer is not limited by the procedure boundary, but extends into the global data space. The use of large numbers of "Use" clauses again means that the programmer has to work hard to follow exactly what is happening in the software, and where data and procedures are defined. Of course, some development environment features such as data browsing help to maintain control of some of the above problems, but they cannot alleviate the problems altogether.

Some of the modules in the DIINS software are very large. Industry standard rules-of-thumb for module sizes range between 50 and 100 lines of code per module. In the case of DIINS three modules were measured with PC-METRIC as having 766, 790 and 855 lines of code, respectively, indicating that the modularity of the code could be improved by using more routines (procedures/functions) within the software. At the time of the original study, the DIINS software consisted of approximately 372 routines.
The main point of the above is to indicate that, because of the evolution that has taken place in the software, and the design approaches adopted, the design of the DIINS XDM code does not

promote maintenance, including modification to produce the ADM software. Hence, care will have to be taken during the re-engineering of the software to ensure that the maturity that exists in the software currently, in terms of an established low defect (bug) rate, is not significantly degraded during the modification process. The defect rate will be affected in two ways during the modification:

a. Re-engineering enhancements (made with hindsight) should improve the design and robustness of the software; hence, should lead to a lowering of the defect rate.

b. The fact that the code is being modified significantly will lead to an increase in the defect rate.

The main aim when performing these two processes (improving the design and extending the design) should be to disturb the software as little as possible, and ideally the goal is to have the net defect rate either unchanged or improved.

In spite of the identified shortcomings, the DIINS software is well designed software, albeit a functional rather than "modern" Ada object-oriented design. A well thought out tasking architecture has been put in place, and the manner in which the Kalman Filters and Failure Detection mechanisms have been designed and implemented, with extendability in mind, should greatly simplify the task of satisfying some of the ADM requirements, e.g. extending the number of filters to be handled to 15. In general the choice of module and variable names in the software is good, significantly enhancing the readability of the software, which is an important issue when it comes to modifying the software.

3.3.2 Current Status Of The XDM Software

Since the DIINS Software Enhancement Study was conducted, DREO have performed additional work on the XDM software aimed at increasing its portability, in particular, with a view to easing the transition from the XDM to ADM environment. In particular, dependencies on proprietary software components in the XDM have been reduced. Areas addressed include the removal of the dependency on Telesoft mathematical functions. Some mathematical function dependencies still exist, but it is anticipated that these can be removed relatively easily when the need arises. In addition, the proprietary mail system has been replaced by portable code developed by DREO.

3.3.3 Enhancements For The DIINS ADM Software

3.3.3.1 General

With the above discussion of the DIINS XDM software in mind, the PRIOR plan when enhancing the XDM software to produce the ADM will be to focus on the software changes that specifically cater to the new requirements called for by the RFP. No attempt will be made to put into effect all of the recommendations coming out of the DIINS enhancement study. However, where it is considered appropriate during the project (and in selected areas) to implement the study recommendations this will take place. Any changes made will be well documented and tested during the project, to ensure that the maintainability of the software will be significantly enhanced over that of the XDM software.

PRIOR has reviewed the changes required for the DIINS ADM, and **Figure 3.1** shows the interfaces involved for the ADM.

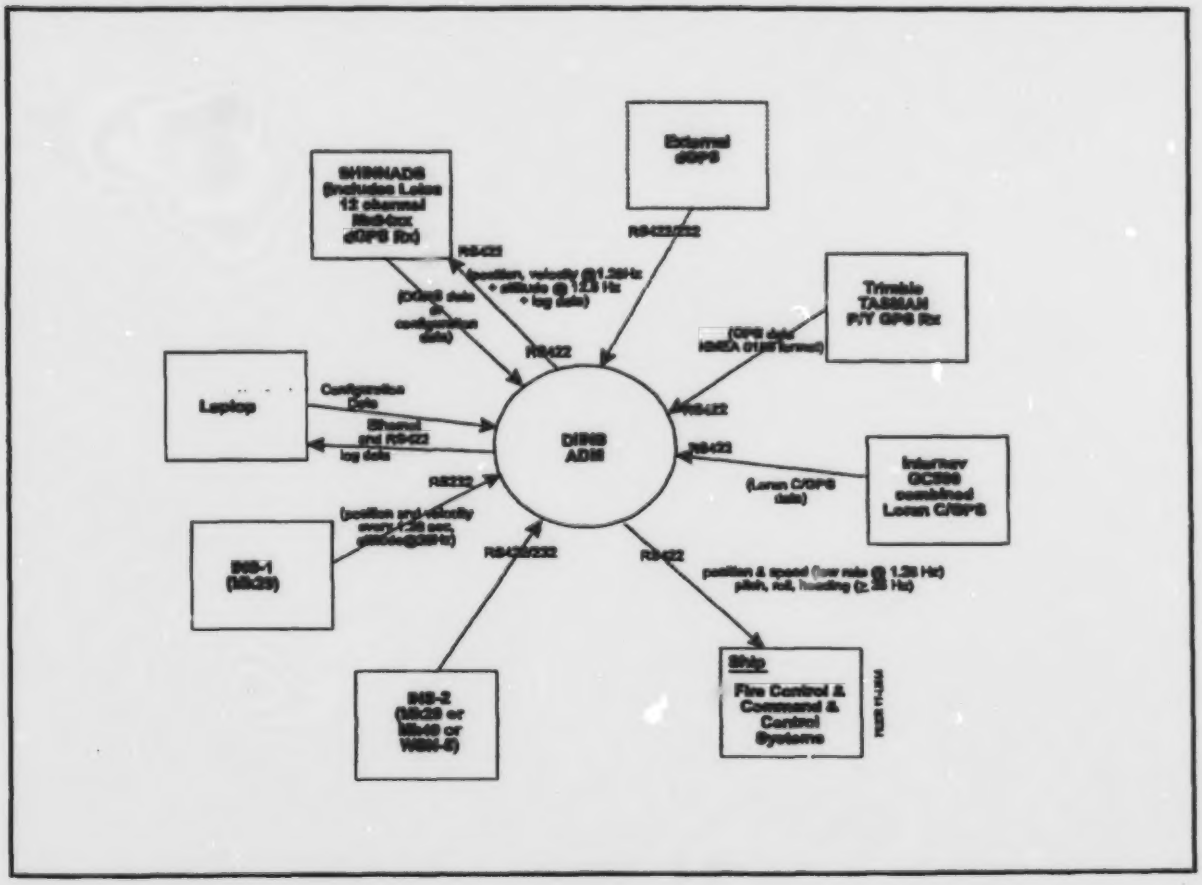

Figure 3-1 DIINS ADM Context Diagram

Some of the most significant areas where changes will be required to the DIINS software are as follows:

a. The introduction of two-way communications between DIINS and SHINNADS to allow display and modification of the sensor and filter internal parameters;

b. The use of up to 15 Kalman Filters (the exact number of filters is to be confirmed during the development testing), instead of the 7 filters present in the XDM version of DIINS that underwent study in the DIINS Software Enhancement Study;

c. Distribution of the Roll, Pitch and Heading at a minimum of 25 Hz;

d. Processing of the dGPS input in real-time;

e. Making the sensor interface to DIINS more amenable to varying the number and type of sensors interfacing to DIINS;

f. Data logging to provide data from sea trials to undergo shore based analysis; and

g. The provision of modern Human Computer Interfaces (HCIs) to DIINS, both from the SHINNADS display and via a portable PC interface.

A number of the DIINS enhancements indicated above require both hardware and software changes to DIINS, and in some cases the extent to which the requirements will be satisfied by hardware (e.g. in the case of high update rate data distribution) has been examined as part of the systems analysis conducted during the Phase 1 study, and is discussed elsewhere in this report. The following sections discuss the changes to the DIINS XDM software in more detail.

3.3.3.2 Sensor Interface Software Enhancements

The issue of the adaption of DIINS to other sensors was given careful consideration during the DIINS Software Enhancement Study (Section 7 in Reference[4]). For convenience the content of Section 7 of the original study report has been included as Appendix C to this report. In the analysis recorded in Reference [4], consideration was given to how the DIINS interfaces, both input and output, could be enhanced by adopting an alternative software architecture. As a result of the analysis conducted it is proposed to modify the current DIINS sensor interface into a more generic (general and in the Ada sense) interface. It was recognized during the original study that there is currently a high degree of commonality between the processing for each sensor at the interface to DIINS, and that there are some differences between the processing required for each sensor. These differences could, for example, be handled by passing functions catering for the differences into the Ada generics. Other issues addressed during the analysis were the parameterization of the sensors and other design aspects aimed at making it easier to reconfigure DIINS to adapt to new sensors. During the DIINS ADM systems analysis, the results from the study have been reviewed in light of the ADM requirements, and against the project cost and

schedule constraints. The following discusses the system design decisions in the area of enhancements to the sensor interface.

The XDM software has been well designed in the area of the Kalman Filtering to allow for the number of filters to be increased, and no significant issues on the software side are anticipated in introducing this change to the software. Indeed, the XDM has already been run in a 15 filter configuration, albeit at a reduced update rate, i.e. filter outputs every 3 x 1.28 seconds (yielding 99% processor load). A figure of 4 x 1.28 seconds update rate would probably be more appropriate to add some processor margin. The main impact of this change to the system is in the amount of processing power required from the DIINS processor to run the filters in real-time.
One area where improvements to the sensor interface have been identified by the DREO staff during the XDM development is that the configuration parameters would be better grouped along sensor lines (files) rather than the current grouping of all of the sensor configuration information into a single filter configuration file. Where possible during the ADM development the sensor information will be grouped together to promote ease of addition/removal of sensors in the future, by making it possible to add (or remove) all of the information associated with a sensor via a single sensor configuration file for the sensor, see Appendix I for details of the proposed ADM Configuration Data organization.

The sensor interface enhancements will be based on the architectural changes proposed by PRIOR during the DIINS Software Enhancement Study. Appendix C to this report contains details of the proposed changes for the ADM in the area of the sensor interface. Appendix C is based on the content of Section 7 in Reference [4], with changes made as appropriate to indicate the changes to be made to the XDM software architecture during the DIINS ADM development.

3.3.3.3 Data Input-Output Interface Software Enhancements

The Input-Output interface enhancements will be based on the architectural changes proposed by PRIOR during the DIINS Software Enhancement Study. Appendix C to this report contains details of the proposed changes to be made to the XDM software in the area of the data input-output architecture to create the ADM.

3.3.3.4 High Update Rate Data Distribution

The digitized attitude (roll, pitch and heading) data from the INSs will be redistributed to the other ship systems via an RS-422 (or possibly via a LAN, e.g. Ethernet) interface in the DIINS ADM. An interpolation procedure will be used to determine attitude information between the sensor sampling points, to allow the attitude data to be distributed at a rate higher than the sensor sampling rate, yielding an attitude distribution rate >=25Hz. In the case of the Sperry MK-49 INS, it is planned (by DND) that the interface to DIINS will be the same as that of the MK-29 INS, including the data distribution frequency.

Before redistributing the information to the other ship's systems, e.g. Gun, the high rate data will be failure detected and corrected based on the DIINS ADM filter outputs. The details of the corrections to be applied by the ADM to the attitude data before redistributing the data will be defined during the detailed design phase of the project.

It is envisaged (by PRIOR) that in an actual application of the distributed attitude data in a command and control situation, the recipients of the attitude data would be required to perform further processing of the information received from DIINS, in order to correct for the time delays from sensor data sampling to the time of the use of the data, e.g. as attitude information for gun firing.

3.3.3.5 Processing of dGPS Data In Real-Time

3.3.3.5.1 General

Although the XDM has not been operated with real-time dGPS data, it has been run with pre-recorded data. Hence, the ability of the XDM to process dGPS data has been proven to a large extent. The main impact of the introduction of the dGPS sensor into DIINS is the potential impact it has on the number of filters that have to be run for failure detection purposes. In the worst case the number of filters could increase to 15 from the value of 7 with a single GPS.

3.3.3.6 Data Logging

3.3.3.6.1 Introduction

As with the sensor interface analysis conducted in the previous DIINS XDM study, the study also addressed the issue of the DIINS data logging requirements, and made some recommendations as to improvements needed in the area of data logging. As part of the DIINS ADM analysis, the results of the previous analysis (Section 6 in the Ref[4] study report) have been reviewed in light of the DIINS ADM requirements and constraints. Further analysis of the DIINS ADM Data Logging requirements will take place during the development of the ADM.

In general, as the DIINS XDM already contains data logging, albeit to development environment disk and tape, rather than via the operational DIINS ADM equipment, Data Logging for DIINS ADM is not considered to represent a significant problem. The most significant aspects of the design analysis have been the identification of the data to be logged, the definition of the data formats for logged data transmission and the development of a scheme to allow operator selection of the data to be logged.

This section reviews the data logging function performed by the DIINS XDM. It reviews the data being logged for completeness for post-operation analysis.

Post-operation analysis should allow the knowledgeable user to do the following:

- Analyze DIINS accuracy for a particular DIINS configuration;
- Analyze DIINS FDIR function actions; and
- Identify hardware problems and/or embedded software bugs.

If DIINS exhibits unexpected behavior, then the user should be able to record operational data and send it to the DIINS manufacturer for analysis. This analysis can include the following steps:

- Review of the log file that lists the DIINS activities. The log file will indicate hard failures and FDIR actions.
- Review of the DIINS Kalman filter data generated by each of the DIINS filters.
- Review of the data generated by each navigation sensor. In particular, the analyst may compute differences between navigation sensors in order to identify anomalies.
- Computation of a post-processed DIINS solution. Ideally the post-processing software exactly reproduces the real-time software functionality, so that a bug that occurred during real-time operation will also occur during post-processing.
- Computation of a reference navigation solution using an enhanced post-processing software package that includes non-causal smoothing. The reference solution is the most accurate available from the post-processed data. The analyst then computes a difference between the real-time navigation solution and the reference solution, and compares the difference time history with the real-time Kalman filter RMS estimation uncertainty.

The data logged by the DIINS ADM should support all of the above steps.

To summarize, the DIINS XDM appears to provide data logging that meets most but not all of the data logging requirements indicated above. The subsequent sub-sections provide detailed explanations.

3.3.3.6.2 Recorded Data Requirement

The following sections indicate the data that the DIINS ADM should record to support comprehensive post-operation analysis, together with a commentary on what the DIINS XDM currently records.

3.3.3.6.2.1 Log File

The DIINS ADM should record all events and actions. This log should include the following:

- All messages generated by the navigation sensors which are received by DIINS.
- All actions taken by the DIINS, in particular, each FDIR action and message to the operator.

- The log should include a listing of the DIINS configuration parameters used for the given run. The parameter list should be repeated every time the DIINS ADM is re-started, on the assumption that the parameters may have changed.

The DIINS XDM data logging capabilities meet the above data logging requirements.

3.3.3.6.2.2 Raw Sensor Data

The digital data is available from the DIINS sensors at predefined sampling rates. The DIINS XDM records all of the raw sensor data that it receives. Every 1 second all the data generated by the GPS receivers and the Loran C receiver are received and logged. The XDM records the position and velocity data generated by INS1 and INS2 every 1.28 seconds, and the attitude data generated by INS1 and INS2 every 40 ms (25 Hz).

Ideally, the data logging should record all status information that a sensor generates. This would allow the analyst to review the sensor status independently of the interpretation of the sensor status generated by DIINS. This requirement must be weighed against the requirement to operate within the constraints of the status information received from the sensors, e.g. only with the status information in the NMEA messages received from the GPS. The NMEA message set may not provide all status information that, for example, the particular GPS receiver can output. The XDM records all sensor status information that it receives from the sensors.

The DIINS ADM will log the same raw sensor data and sensor status information as is logged in the XDM.

3.3.3.6.2.3 Kalman Filter Data

The DIINS should record the data generated by each Kalman filter.

The data for each filter comprises time histories of the following data items:

- Estimated error state vector;
- RMS estimation uncertainty vector; and
- Measurement residuals vector and RMS residuals vector.

The DIINS XDM logs all of the above data associated with the filters.

The logged data could include the complete estimation error covariance matrix in place of the RMS estimation uncertainty vector. It could also include the constructed measurements vector and the measurement noise standard deviation vector. These data items are not part of the data currently logged for the XDM, but they may be considered for the ADM development depending on the approach adopted to post-operation analysis for the DIINS ADM. The exact details of the

filter data to be logged in the ADM will be determined during the detailed design phase of the project.

3.3.3.6.3 XDM Data Logging Log Capacity

The details of the data logged in the XDM are shown in Table 3.3.3.6-1. For details of the data logged see Appendix E.

Table 3.3.3.6-1 XDM Logged Data

• Raw Sensor Data:		
• GPS	189	1
• dGPS	189	1
• INS1	800	1.28
• INS2	800	1.28
• Loran	96	1
• Filter data for 7 or 15 filters:		
• Display Data	1560	1.28
• State vector	5056	1.28
• Residual & Variance	1896	1.28

Based on the sea trials conducted previously for the XDM, the volume of data logged in two hours to EXABYTE tape was 22MByte (for 7 filters (sensors were 2 x INS, GPS and Loran), uncompressed). Over a 24 hour period, this implies that approximately 264 Mbyte per day would be logged for 7 filters. As indicated in Appendix E, the maximum data logging capacity to log data for the ADM will be 724 MByte/Day uncompressed.

It is assumed for the data logging that permanent logging on the various ships fitted with DIINS ADM would be provided by logging to the hard disk on the portable PC attached to the system. Portable PCs are available with 1GB+ of removable hard drives; hence, it should be possible to get more than two days of data onto such a hard drive. This appears to be the most cost-effective manner in which to provide the data logging storage for DIINS at all locations where it is fitted.

Other alternatives are to include storage devices in the VME chassis (e.g. Red Rock Technologies provide 4.2 GB of hard drive in 2 VME slots, and 8.5GB in 3 VME slots). However, the approach of adding extra storage capacity into the VME chassis is a problem, as there is a shortage of slots to use, and the DIINS ADM operational requirement does not warrant the addition of such fixed storage capability. Another alternative for extra mass storage would be for the maintenance engineer to carry a portable storage unit (e.g. EXABYTE tape drive) that can be plugged into a SCSI port on the DIINS ADM. The delta cost in this option is that of the purchase of a single EXABYTE drive. This is the preferred option from a practical viewpoint to support maintenance, as it saves having to remove the hard drive from the portable/fixed system on the ship to be sent for on-shore analysis; instead, an EXABYTE tape would be sent for analysis, as is currently the case for the DIINS XDM.

Note: The DIINS XDM currently uses two EXABYTE tape drives included in the VME chassis to perform data logging, and these drives were used to log data during the last sea trials. The VME chassis also contains a Winchester disk drive, but this is not currently used.

Conclusion

In general the DIINS data for data logging will be put onto the portable PC hard drives. A number of log files will be overwritten cyclically, so that when an anomaly occurs data is available around the time of occurrence of the anomaly to be analyzed ashore. A portable EXABYTE drive (with a SCSI interface to DIINS) could also be used by the maintenance engineer and during sea trials to log data for on-shore analysis.

3.3.3.6.4 ADM Data To Be Logged

In general, the data to be logged in the case of the ADM will be the same as was logged in the XDM, see Appendix E.

3.3.3.6.5 Data Logging Operator Selections

See Figure D-8 and its associated text in Appendix D for details of the logging data selections assumed to be available to the user in the DIINS ADM.

3.4 System Components

3.4.1 Introduction

The existing system components consisting of the SHINNADS display system and its internal VME chassis will be used to house the DIINS ADM. The requirement for upgrading the processor and interface cards has been examined during the Phase 1 study.

Based on the results of the Phase 1 analysis, it is assumed that the processor board will likely change to a 68060 board for the ADM, but that the input-output serial interface board will remain the same as it will meet the ADM requirements. For the present it will be assumed that the card frame requirements for the ADM will be basically the same as for the XDM. This means that three VME slots should be reserved for circuit boards and additional room should be reserved for the transition module for the serial input-output board. In practice, if a move is made to the 68060 for the ADM it should be possible to occupy just two VME slots plus the transition module position, as the processor board would contain the PROM memory currently met in the XDM by the addition of the MVME 216 memory board.

As DIINS just fits into VME slots in the SHINNADS chassis, there is no urgency to fix the exact configuration of the boards, so for now all that is important to do is to reserve sufficient space for a decision to be taken later. In the same way there is no urgency to select between the 68040 and the 68060 processors, as additional testing needs to be conducted during the ADM development to firm up on the processor selection. Based on price alone, as will be seen later in this report, the 68060 processor is favored over the 68040 processor, so it is the likely to be the processor selected.

There are alternatives to the Force SYS68K ISIO-2 Input-Output board used on the XDM. In some cases daughter boards can be added to processor boards to provide the input output capabilities. The CDC VME based Pentium (Cheetah P5, 133Mhz , 64 MB, -40 deg/+85 deg C) board comes fitted with six software configurable RS-232/422 serial interfaces (board price $ 19,938 CA). The DIINS ADM requires eight serial interfaces.

The two main advantages of remaining with the XDM serial interface solution are that it minimizes software modifications required for the ADM, and it provides an additional I/O processor within DIINS, which could be used to support the high rate attitude data distribution requirement if this turns out to be the best solution.

3.4.2 Processing Component

Motorola's newest high-end CISC processor, the 68060, offers two to three times the performance of a 33Mhz 68040, while maintaining backwards compatibility with the millions of lines of existing 68000-family code. This is the latest and most powerful member of the 68K family. It uses Superscalar pipelined architecture which means it can perform more than one instruction at a time. The 68060 allows simultaneous execution of two integer instructions (or 1 integer and 1 float instruction) and one branch during each clock cycle. The 68060 has clock speeds of 50 and 60 Mhz currently with a future model at 66 Mhz .

Moving applications from a 68040 environment to a 68060 environment is relatively straightforward, provided the designer identifies and adapts for system and chip level hardware changes. The task of converting an 68040 application to run on the 68060 platform can range from trivial to moderate in difficulty. The job comes down to converting software to run, both on the new processor and with the peripheral hardware on the new board.

Because the 68060 is largely backwards compatible with the 68040, especially in terms of user-mode code, porting to the new chip poses only minor problems. The programmer can recompile using an 68040 compiler with a Motorola-provided 68060 library which compensates for differences in the instruction set. Alternatively, the programmer can use one of the emerging 68060 compilers. Some further investigations will take place during the software design phase as to the best compiler to use for the ADM development.

Table 3.4.2-1 shows the Millions of Instructions Per Second (MIPS) figures taken from various Motorola promotional literature documents and indicates the performance gains of the various generations of the 680x0 family.

Table 3.4.2-1 MIPS Relationship to CPU Type and Clock Speed

Clock / CPU	8 MHz	16 MHz	20 MHz	25 MHz	30 MHz	40 MHz	50 MHz	66 MHz
68000	1.2							
68EC000	1.2	2.5						
68020		4.3	5.2	6.5	8.7			
68EC020		4.3		6.5				
68030		4.5	5.4	6.7	9.0	10.8	13.5	
68EC030				6.7		10.8		
68040				26.1	34.8	41.8		
68LC040			20.9	26.1	34.8			
68EC040			20.9	26.1	34.8			
68060								100

One of the main advantages of staying with the 680x0 family of processors for the DIINS ADM is that the existing software development and support environment can still be utilized, since the code is backwards compatible.

3.4.3 Motorola MVME 177-004 Processor

The most likely processor to be selected for the DIINS ADM is the MVME 177-004. The main features of the proposed MVME 177-004 processor are as follows:

- 50 Mhz MC68060 32-Bit Processor with MMU and FPU
- Full 32-Bit Master/Slave VMEbus interface
- 32 MB On-Board DRAM
- 128KB SRAM with battery backup
- On-Board SCSI Interface
- On-Board Ethernet Interface
- 4MB of FLASH ROM
- Two 44-pin sockets for up to 2 MB On-Board ROM/EPROM
- Four EIA-232-D Serial Ports
- Centronics-Compatible Parallel Port
- Four 32-Bit timers, and one Watchdog timer
- 8KB of NVRAM with Real-Time Clock/Calendar
- Remote Reset/Abort/Status/Control Functions

3.4.4 Force SYS68K ISIO-2 Serial Interface

The DIINS ADM serial interface will be via a Force SYS68K ISIO-2 board. This is an intelligent Serial I/O board that includes 8 programmable high speed serial ports. The ports are fully configurable as either RS232 or RS422, and an on-board microcontroller alleviates much of the I/O processing from the main DIINS CPU board.

3.4.5 Force SYS68K ISIO-1FP Serial Interface Transition Module

The DIINS ADM will use a Force SYS68K ISIO-1FP board to interface the external system components to the DIINS serial interface connectors. This front panel transition module contains eight 9-pin Sub D connectors for the eight serial ports from the ISIO-2.

3.5 Processing and Memory Capacity

3.5.1 Processing Requirements

3.5.1.1 DIINS XDM Processor

With 7 filters operational and outputting the processed position information at the highest rate (1.28 sec) the DIINS XDM uses 85% of the available processing capacity (i.e. 1.088 sec), based on the measurements made using the XDM internal processor utilization measurement facilities. The startup phase of the processing uses more processing power; therefore the above represents the steady state requirements.

For the 15 filter processing case, with the filtered position output being generated every 3*1.28 sec (i.e. 3.84 seconds), there is <1% spare CPU capacity. Hence, the 15 filter algorithms take approximately 3.8 seconds to execute on the XDM.

The above indicates, as expected, that the performance requirement increase in going from the 7 to the 15 filter case is not linear, and calls for an increase in processor performance of a factor of 3.5 (based on 3.8/1.088) to achieve the 85% utilization (i.e. 1.088 sec processing time) with a data output rate of 1.28 sec for the ADM with 15 filters. This is still a high processor utilization. To bring the utilization down to a level with more margin, e.g. 70% utilization (i.e. 0.896 sec processing time), would need an increase in processor power of a factor of 4.2 (based on 3.8/0.896) over that of the XDM.

3.5.1.2 DIINS ADM Processor

The PRIOR proposal assumed a factor of two to three times the performance would be achieved in going from the XDM to the ADM processor if the transition was from a 68040 to a 68060 processor. With an actual factor of 4.2 required to run a 15 filter ADM configuration, the 68060 would not have sufficient processing capacity to meet the needs of the ADM, if a 15 filter configuration updating every 1.28 seconds was assumed. In order to assess the capabilities of some of the more modern processors to satisfy the DIINS performance requirements, the PowerPC option was examined in detail as part of the systems analysis conducted. The results of this analysis are discussed in this section. An alternative to the PowerPC, not considered from a technical viewpoint in this report, is the INTEL Pentium processor on a VME board. The performance of the PowerPC and Pentium options would be basically the same.

One option for addressing a significant increase in processing requirements for the ADM is to move away from the current 680x0 series processor to the more modern PowerPC. **Table 3.5.1.2-1** compares a number of parameters associated with two 680x0 series boards (the MVME 167, 68040 based (current DIINS processor)) and the MVME 177 (68060) board with two PowerPC boards, one from Motorola (the MVME 2600) and one from FORCE Computers (the PowerCore). Motorola has also released a 2700 PowerPC board, with a clock frequency of 233 MHz.

Table 3.5.1.2-1 680x0 v. PowerPC Processor Board Comparison

Clock Freq (Mhz)	25/33	50	200	120/166/ 200
Cache (KB)	8	8	Level 1 16/16 or 32/32 * Level 2 256	Level 2 up to 1 MB
MMU	Y	Y	Built in	Built in
DRAM (MB)	4 to 64	4 to 256	16 to 256	16 to 272
SRAM (KB)	N	128	N	N
SCSI I/F	Y	Y	Y	N
Ethernet	Y	Y	Y	Y
Floppy	N	N	Y	N
Mouse	N	N	Y	N
Keyboard	N	N	Y	N
E-PROM/FLASH ROM (MB)	Up to 4MB ROM/ EPROM	Up to 2MB ROM/ EPROM plus 4 FLASH	1 Socketed 4 to 8 Surface Mount	Up to 8
RS-232 Ports	4	4	4	1
Centronics I/F	Y	Y	Y	N
Watchdog Timer	Y	Y	Y	?
32 bit timers	4	4	N	?
16 bit timers	N	N	4	?
Power Consumption (Watts)	<20	<20	25-30 *	14 - 25.5 **
MIPS	36.8 (33Mhz)#	56 (est. based on 167)	N	N
SPECint95 (estimated)	N	N	5.2 to 8.7 *	7.4 **
SPECfp95 (estimated)	0.99 (est. ##)	1.5 (est. ##)	3.7 to 8.5 *	5.8 **
PMC Slot	N	N	Y (1 slot)	Y (2 slots)
MTBF (hrs) Mean/90% Confidence	?	?	190,509 / 107,681 @	35,000 to 38,000 excl daughter boards
FP Arith Capabilities	FPU	FPU	Built In	Built In
Vibration (10 to 150 Hz)	?	?	?	5g

Table 3.5.1.2-1 680x0 v. PowerPC Processor Board Comparison

Shock (11ms half sine, 6 axis, 3 pulses)	?	?	?	15g
On-board Debug Mon & Self Test	?	?	Y	?
Warranty	?	?	5 year limited	?

Notes on Table 3.5.1.2-1:

*: Depends on the processor chip selected, MPC603 or MPC604

**: PowerPC 603e (120 - 166 Mhz) ,PowerPC 604e (200 Mhz)

@: Based on sample testing in accelerated stress environment

PMC: PCI Mezzanine Card Slot, for adding additional I/O interfaces.

SPEC performance figures measure the performance of the processor, memory hierarchy and compiler relative to the performance of a SPARCstation 10/40 (40MHz SuperSPARC processor with no L2 cache). Note for comparison, a 200MHz 256K/L2 Pentium Pro has SPEC figures of SPECint95 = 8.2 and SPECfp95 = 6.21, while an UltraSparc-200MHz has SPEC figures of SPECint95 = 7.7 and SPECfp95 = 11.1 .

#: For comparison, a SPARCstation 5/110MHz is rated at 135.5 MIPS, whereas the UltraSPARC 200MHz CPU is rated at 414 MIPS (136 MFLOPS, SPECint92 = 332 for 2 CPUs, SPECfp92 = 505 for 2 CPUs).

##: For simple comparison the figures for the UltraSPARC have been used to estimate the SPECfp95 data, i.e. for the 68040 = (36.8/414)* 11.1 = 0.99 SPECfp95. For the 68060 card, at 50mhz we would have approximately (56/414)*11.1 = 1.5 SPECfp95.

As can be seen from Table 3.5.1.2-1, based on a comparison of estimated SPECfp95 performance figures for the current 68040 processor and a 200Mhz Motorola PowerPC card, the increase in performance going to the PowerPC would be a factor of 8.6.

3.5.1.3 Development Environment

If a PowerPC were selected to satisfy a 15 filter requirement, it would make sense to change to VxWorks as the runtime environment, and the VADS or the Green Hills development environments. The only problem is these environments are expensive. The price of a change to the development environment was not budgeted in the ADM development costs. If the cost of such a development environment were spread over the production phase of the ADM project, then the combined cost of the hardware and software environment for the production of the anticipated 16 units could be cheaper than staying with the current expensive 680x0 processor boards. The only problem is that the development environment is needed in the development, and not the production phase; hence, unless some of the production phase budget can be moved into the development phase to cover the cost of development, this would not be a practical proposition.

3.5.2 Memory Requirements

3.5.2.1 PROM

The program for the OPVAL configuration of DIINS XDM (2 x GPS, 2 x INS, and Loran (no Omega or Speed Log), network display, but no data logging) requires about 0.99 MB PROM to hold it in. The "Full Blown" version of DIINS (including data logging to tape and the Speed Log) requires approximately 1.23 Mbyte of PROM. A 2 Mbyte PROM should be adequate for the DIINS ADM requirement.

3.5.2.2 RAM

In the XDM system, the MVME 167-34B card is fitted with 32 MB of RAM. It is not known how much of this RAM is actually used in the XDM, but it is known that this is sufficient to run both the 7 filter and 15 filter case. It is envisaged that 32 MB of RAM should be adequate for the DIINS ADM.

3.6 Cost

3.6.1 Cost Comparisons Of ADM Hardware Solutions

This section compares, see **Table 3.6-1** and **Table 3.6-2**, the costs of a number of hardware solutions examined for the DIINS ADM Implementation during the Phase 1 study, namely:

- Existing XDM hardware configuration;
- Configuration based on 68060 processor; and
- Configuration based on PowerPC processor.

The cost of the solution to the DIINS ADM hardware configuration will depend on the performance it is required to be achieve for the filtering. If the filtering requirements are high, in terms of the number of filters to be run in parallel and the number of filter states, then the 68060 processor will not be adequate, and a switch to the more powerful PowerPC series would be required.

The original PRIOR proposal assumed that synchro-to-digital resolvers would be used to convert the attitude data from the INSs to digital form in order to support high rate attitude data distribution to the other ship's systems. As a result of the discussions held with DREO during Phase 1 of the project, it has been agreed that the synchro outputs will not be used, and instead the digital outputs from the INS will be distributed to demonstrate the high rate data distribution during the DIINS ADM project. This means that the costs indicated in this section no longer indicate the cost of synchro-to-digital conversion hardware.

Note: Prices indicated in the tables in this section are list prices, i.e. exclude volume discounts and markups, and exclude PST and GST, shipping and warranty. Volume discounts will not be high for the 16 off units required for the DIINS ADM, and for the Motorola Processors examined discounts do not apply. The boards are manufactured in the United States, hence the Canadian prices shown will vary with the exchange rate. The exchange rate used in the currency conversions for this report were 1$ US equals approximately $1.43 Canadian. The cost of the VME chassis for the DIINS ADM is assumed to be covered in the cost of the SHINNADS system. The hardware costs shown are for the hardware as manufactured, and exclude any additional costs that will be incurred during the ADM implementation phase, e.g. cabling, installation, test, documentation, QA, Project Management and labor in general.

Table 3.6-1 Option 1: 68040 Based DIINS ADM Solution (same as XDM solution)

1	Motorola MVME 167-34B	33MHz 68040 single board computer with 32 MB RAM.	4	16,876.80
2	Motorola MVME 216	2 MB EPROM and 2 MB Static Ram.	4	2,606.48
3	Force SYS68K ISIO-2	Intelligent Serial I/O board that includes 8 programmable high speed serial ports. The ports are fully configurable as either RS-232 or RS-422, and an onboard micro controller alleviates much of the I/O processing from the main DIINS CPU card.	TBD	2,800.00
4	Force SYS68K	Front Panel Transition Module	TBD	520

Table 3.6-1 Option 1: 68040 Based DIINS ADM Solution (same as XDM solution)

	ISIO-1FP			
Total				$22,803.28

Table 3.6-2 Alternative DIINS ADM Processor Board Solutions

2	Dynatem DPC1 (DRC1)	Pentium VMEBUS SBC 100Mhz Pentium with 32 MB DRAM (Ruggedized Version, -40 to + 85 deg C)	TBD	5,641.35 (10,314)	8,961.35 (13,634)
3	Force 511070/ Force 511076	PowerCore-6603e/16-120-L0-4 - 120 MHz PowerPC 603e CPU Board with 16 MBytes DRAM and 4 MBytes user flash, 10Base-T Ethernet interface with MicroMon firmware in boot flash. High performance VME64 to PCI interface. 2 PMC slots and 1 serial I/O port. Includes installation guide. Plus: PowerCore-MEM/64L - 64 MBytes memory expansion module for the PowerCore Products. For use in lower expansion slot.	12-14 Plus 2-4	4,472.00 + 2,560 = 7,032	10,352
4	Force 510916	CPU-60D/32 - 50 MHz 68060 CPU Board with 32 MBytes shared DRAM, 4 Mbytes Sytem Flash, Ethernet/SCSI-2/Floppy interfaces, 2 serial I/O	12-14	10,280.00	13,600

Table 3.6-2 Alternative DIINS ADM Processor Board Solutions

		channels, VMEPROM, User Installation Guide			
5	MVME 177-004	68060 50Mhz, 32 MB	TBD	12,835	16,155
6	MVME 177-014	68060 60Mhz, 32 MB	4	17,327	20,647
7	MVME 177-015	68060 60Mhz, 64 MB	13	24,750	28,070

Note:
The cost of the RS232/422 input/output boards are as indicated in Table 3.6.1 for Slots 3 and 4, i.e. $3,320.00.

As indicated above, the cost of the hardware to implement a unit of the DIINS ADM changes significantly depending on the choice of the processor. As can be seen from Tables 3.6-1 and 3.6-2, the cheapest solution examined was Option 2, which was Pentium 100 Mhz based and cost $8,961 CA for the circuit boards. The cheapest Motorola based solution, based on the same processor range to the XDM (hence minimizing risk) was Option 5, which is 68060 50 Mhz based, costing $16,155 CA for the hardware. If the ADM development takes place on the 680x0 family of products, then the DIINS XDM development environment will be able to be reused (e.g. for migration to the 68060 from the 68040). This approach to the software enhancement should be the minimum risk scenario. The DREO staff have already performed some experiments aimed at migrating to the 68060, and some issues have already been identified, such as compilation issues. However, no significant effort has been devoted to this activity to date, and there is no reason to suppose that when the appropriate level of effort is applied that the port to the 68060 cannot be achieved relatively easily.

Table 3.6-2 presents the cost of the 680x0 processor from Force computers together with the cost of the Force PowerPC processor. Although the cost of the Force 680x0 series SBC is lower than the Motorola version, changing to the Force SBC would represent an increased risk over using the Motorola MVME 177, due to the differences between the two boards from the two vendors, even though they use the same processor family.

3.6.2 Alternative Software Development Environment

If a change were made to use a different processor family, e.g the Pentium or the PowerPC, then the cross-compiler will need to be changed to target the software at the new VME board. This essentially means that if the processor family were changed, the software development environment will have to be changed. For Ada development on the Pentium or PowerPC, the GreenHills AdaMULTI environment could be used for a host development (e.g. compiler, debugger etc.) in conjunction with the WindRiver Systems Tornado/VxWorks Host/Target environment. The Tornado environment consists of the following:

- Tornado suite of tools, a set of cross-development tools and utilities on both the host and the target;
- VxWorks run-time system, a high performance scaleable real-time operating system that executes on the target processor; and
- A full range of communications software options, such as Ethernet, serial line, ICE or ROM emulator for the target connection to the host.

The GreenHills compiler and the Tornado environment are designed to work together. The Ada Multi SDE consists of:

- An Ada 95 cross-compiler, targeted at the PowerPC or the Pentium (or the 680X0 series).
- A powerful debugger for use for Ada cross development (i.e. host-target configuration). The debugger includes features to support the debugging of multiple Ada tasks.
- GUI based profiling tools for performance, call graph and code coverage on user application code.
- Ada runtime system that includes the full Ada tasking constructs while benefiting from the hard real-time VxWorks kernel and services.

With the AdaMULTI/Tornado/VxWorks combination, development can take place on either Sun or Windows NT platforms, with the code targeted to the embedded PowerPC/Pentium VME target.

The only drawback with the software development environment indicated above is its price. Table 3.6-3 indicates the cost of the software components required to support development and to support the DM Implementation Phase, e.g. the production run of 16 units. For the implementation phase each hardware unit would be required to contain a VxWorks Runtime license. The pricing shown in Table 3.6-3 assumes that two sets of the development environment would be required to undertake the DIINS ADM development.

Table 3.6-3 Possible GreenHills/WindRiver Systems VxWorks Based Solution For The ADM

AdaMULTI SDE - Seat 1	13,442
Seat 2	12,770
Tornado Dev Env - Seat 1	27,599
Seat 2	4,605
Sub-Total	$58,416
16 x VxWorks Runtime Licenses ($1,168 each)	18,676
Total for 16 units	$77,092
Cost per unit	$4,819

Notes:

- The Additional seats of AdaMULTI up to 5 are $12,770 CA each.
- The AdaMULTI price includes 90 days of technical support.
- The Tornado price includes maintenance.
- An additional seat of the development environment, if required for the ADM Development, would cost $17,375, or the equivalent of an increased price of $1,086 each for the 16 production units.

3.6.3 Equipment Spares

As the DIINS ADM MTTR is 15 minutes on-board ship this implies a complete set of spares is held on-board each ship, unless board commonality with other ship systems exists. These spares have not been considered in this report to be part of the $20,000 target budget for a single unit of the DIINS ADM. Hence, the cost associated with spares to meet a 15 minute repair time would basically double the cost of the hardware required for the ADM. The SHINNADS display system currently uses the MVME 147-SA circuit board; hence, there is no board commonality between DIINS and SHINNADS in terms of either the XDM or ADM processor board.

3.6.4 Cost Observations/Conclusions

The comparison between two of the main options of interest based on the costing analysis is as follows:

Material cost per unit (assuming the SDE cost is spread across the 16 units) is a follows:

- Option 2: Pentium 100 Mhz : H/W = $8,961 + S/W SDE portion= $4,819
  Total: $13,780

- Option 5: 68060 50 Mhz : H/W = $16,155

Cost of materials for 16 units:

- Option 2: Pentium Based = $220,480 (15% cheaper over 16 units versus the 68060 solution)
- Option 5: 68060 Based = $258,480 ($38,000 more expensive)

To these costs would have to be added the other production costs other than for the materials only which are covered in this report, and any spares that are required.

Although the above costs indicate that the cost of materials would be cheaper if a Pentium processor were adopted for the ADM, some/all of this cost may be offset by the increased cost of porting the software to the Pentium processor from the 680x0 environment. The original DIINS XDM study highlighted the fact that the XDM at that time did contain a significant amount of compiler specific code. This situation may have been addressed in part by some of the recent activities at DREO aimed at increasing the portability of the code. Using a modern development environment such as the AdaMULTI environment, together with its multi-task debugging capabilities and code profiling capabilities, could offset the added impact of porting to the Pentium. The Pentium processor should increase the performance of the DIINS software, due to the modern processor architecture and higher clock rate than the 68060 50 MHz.

The only way that the Pentium solution is viable is if funding can be transferred from the production phase to the development phase to offset the cost of the new software development environment that would be required.

The minimum risk solution for the development of the DIINS ADM is to remain with the 680x0 series processor, thus minimizing software porting risks. Unfortunately, this means that the DIINS ADM would be developed using older technology and an obsolete software development environment, rather than using modern hardware and a modern software development environment, including the use of Ada 95.

It would be an advantage to maintain the current DIINS, i.e. serial I/O board, as this is one less change in going to the ADM; hence, represents less risk. This board also contains a processor that can be programmed to perform some processing in parallel with the main DIINS processor. This processing could, for example, be used to support the distribution of the high rate attitude data.

Option 5 is the board configuration recommended for the DIINS ADM development phase under the current project constraints.

3.7 Delivery Lead Times

Covered in Section 3.6 where the information is available.

3.8 Human-Computer Interface

3.8.1 Introduction

The user interface modifications required to DIINS to allow operator monitoring and modification of DIINS parameters from either the SHINNADS display or a portable PC is not seen to represent any significant problems. The main areas where display improvements are required are to provide a clearer indication of the present filter being used, and the current health of all the sensors. In addition, a series of configuration screens is required to provide operator control over the parameters currently fixed within the XDM code. Again, as with the other aspects of the system, the main effort will be in determining the exact requirements and defining the design in sufficient detail for it to be implemented. There will be basically two types of data that is required to be controlled via the Human-Computer Interface (HCI). First, there will be individual parameters to be monitored or set using the interface. To allow this it is envisaged that modern form and menu features will be used to enable these requirements to be implemented. In addition there may be cases where larger volumes of data are to be managed, e.g. in the form of a configuration management file to be input to DIINS or to be modified on site via the HCI. In this case text editing features will be available via the HCI, and the ability to retrieve data from files to be downloaded to DIINS, or save blocks of data received from DIINS to file will be incorporated to meet such requirements.

An important aspect of developing HCIs is to prototype them early on, so that the look-and-feel of the interface can be appreciated and tried by the operators to detect changes needed to enhance the interface. During the Phase 1 analysis PRIOR has prototyped the HCI for DIINS in order to get early feedback from all interested parties. The results of these prototype activities are presented in this section of the report, where the designs of the forms to be available on the user interface are presented.

As there are two user interfaces to be provided (one for the SHINNADS PC display and one for the portable PC interface), it may be possible to make significant economies (and it is also highly desirable from an operator training viewpoint) to make the two interfaces identical, and where possible reuse the requirements analysis, design and if possible the executable code (or portions of it) on the two interfaces. This commonality of the interfaces will be the goal during the DIINS ADM development.

3.8.2 DIINS Operator Roles

In order to analyze the information required to be displayed and modified on the HCI for the DIINS ADM, the details of the configuration files which define the behavior of the DIINS XDM have been examined in detail. The aim of this analysis was to identify the parameters that are appropriate to present to the operator for display and modification.

The information appropriate for display/modification via the HCI depends to a large extent on the purpose of providing operator access to the data. The two key roles of the personnel requiring access to the DIINS data are the operator responsible for the day-to-day operation of DIINS and the maintainer, responsible for tuning the operation of DIINS or for fault finding and repairing DIINS in the event of a failure. It is envisaged that the operator and the maintainer at times may be the some person.

The key difference in the requirements of these two roles is that the maintainer will in general require more detailed access to the DIINS data than will the operator. Hence, from a user interface perspective, the operators interface should be simplified as far as possible, by providing access to only the data required to initialize and run DIINS in a fixed hardware configuration environment, e.g. fixed suite of sensors interfaced to DIINS. The maintainer on the other hand will be required to physically wire DIINS to various sensors as appropriate to create the DIINS installation for the ship. Hence, whereas the operator will want to select which of the available sensors to use in the navigation solutions, the maintainer will want to add and delete sensors from the sensor configuration available to the operator.

Due to the level of detail dealt with by the maintainer, it may be more appropriate for the maintainer to display/modify certain system parameters via a file text display interface, rather than to display all of the parameters involved via custom menus. In such instances it is probably just as quick for a maintainer to set up the configuration using text editing capabilities than it is to use a menu based interface. In addition, the use of custom menu interfaces has to be considered from an overall user interface complexity viewpoint and a cost/schedule aspect of developing custom user interface forms where this is not warranted.

An additional role not considered in the above is the "developer" role. The developer is the person responsible for making the detailed changes to the software in order to modify or extend the capabilities of DIINS.

As the transition from operator -> maintainer-> developer takes place, more detail is required by the person to perform their function. As the detail increases, it becomes less cost effective to place all of the information required for the role on custom forms in the user interface. Indeed, whereas it is desirable in the case of the operator to have all of the necessary data available at the operator interface via custom menus, in the case of the developer, most of the information of interest would not be available via custom forms. The maintainer's data access requirements are somewhere in between those of the operators and the developers.

The current DIINS software is designed to make the addition of new sensors relatively easy. This aspect will be enhanced where possible during the development of the ADM software. It must be recognized, however, that the inclusion of a completely new sensor type, and the development of the appropriate models and software is a complex and time consuming task, and would not be undertaken on-board ship, but rather in the shore-based DIINS software development facility.

Following the precise definition of the roles associated with DIINS in the next section, the other sections will examine the key configuration data in the XDM with a view to determining which roles require access to the information and the manner in which access is proposed to be provided for the ADM, e.g. via a custom forms interface or via a text processing interface.

3.8.2.1 DIINS Role Definitions

Operator (ship-based): Selects and reconfigures DIINS from the available sensors without modifying the physical connections between DIINS and the other ships systems. Operates DIINS in the REALTIME mode only.

Maintainer (ship-based): Installs and reconfigures DIINS by adding and deleting sensors, including modifications to the physical connections, e.g. cabling, between DIINS and the other ship systems. Fault finds failures and performs repairs by board replacement. The maintainer does not, as a rule, make changes to the DIINS compilable software. Operates DIINS in the REALTIME mode only.

Developer (shore-based): The developer modifies the DIINS software to incorporate fixes for software defects or to add/remove features from the DIINS software. May operate DIINS in the REALTIME or SIMULATION (simulated sensors) modes.

3.8.3 Simulation Setup Files

Appendix B contains details of the SimSetup.cfg file used in the XDM. This file contains parameters for the setup of DIINS for both the REALTIME and SIMULATION modes, and shows information related to real and simulated sensors and the file definitions for the case of running 15 Kalman filters simultaneously.

Table 3.8-1 summarizes the parameter types contained in the SimSetup.cfg file, and indicates the roles requiring access to the parameters. In some cases the parameters are assumed to be relatively fixed, and would in the DIINS ADM be set up automatically based on the other inputs that they depend on. For example, the names of some of the files have standard formats, and the only variation will be due to the number of filters being run. Once the number of filters is defined the details of the files can be defined automatically. The more parameters that can be derived automatically from other parameters, the less customization will be required when introducing a new sensor. As can be seen from Table 3.8-1, the majority of the information in

the "SimSetup.cfg" file in the DIINS XDM, could be derived automatically based on the combination of sensors selected for the navigation solution. Hence, in particular, the operator and the maintainer should not be burdened with the task of defining such low level details.

Table 3.8-1 Simulation Setup File Parameter Summary

1	REALTIME/ SIMULATION	REALTIME/ SIMULATION	REALTIME	REALTIME
1	BIG_FILE/ SEPARATE_FILES	Yes	Y	N (Should be selected as separate files by default)
17	Available Sensor Models (e.g. SPERRY_MK29_INS1, SIM_INS1)	Addition of sensor model. Access to all info, including the comms channel parameters (e.g. BAUD rate)	Selection of physically attached sensors available to be installed to support operations	Select from the operational sensors the sensors to be used for navigation
6	Data for all non sensor tasks, e.g. FDI task	Y	N	N
1	Number of filters (Should be derived automatically from the sensor configuration)	N	N	N
1	Simulation time duration	Y	N	N
15	Simulation config file names for each filter's sensor combination (should be derived from available sensors)	N	N	N
17	Sensor data files (SIMULATION mode only)	Y	N	N
15	Filter init file for each filter (fixed function of number of filters)	N	N	N

Table 3.8-1 Simulation Setup File Parameter Summary

	Note: The file for each filter is identical.			
15	Filter output file names (fixed function of number of filters)	N	N	N
15	Filter logging file names (fixed function of number of filters). (Files not used in XDM, assumed will not be used in ADM.)	N	N	N
1	FDI Logging File Name (constant). (File not used in XDM, assumed will not be used in ADM.)	N	N	N

3.8.4 Filter Parameter Files

Appendix B indicates the format of the FOPVALx.dat files, the initialization files for each filter. The details of the format of the file is indicated in the FOPVAL.README information included in Appendix B.

As can be seen from Appendix B there are a large number of parameters associated with the filter initialization. With such a large number of parameters, it becomes questionable as to whether it is easier to use a custom forms interface to modify the parameters rather than just edit the text file using a text editor.

3.8.4.1 Filter Operational Value Initialization Files (e.g. FOPVALx.dat files)

Although Appendix B indicates that there are 15 FOPVAL files, one for each filter, in practice for the DIINS XDM all of these files have identical format and data in them. See **Table 3.8-2** for details of the content of the FOPVAL files. The reason for 15 rather than just 1 file results in part from Ada limitations during the original implementation necessitating the use of multiple files. In addition, it was considered that in future implementations it may be desirable to use different parameters, e.g. update rates for the various filters; hence, the ability in the XDM to set the parameters independently for each filter. In practice, the current FDI software is not designed

to run with filters operating at various update rates, so a single initialization file is probably adequate for the ADM. As it is, when DIINS is initialized and all of the filter tasks commence operation, they read their initialization parameters from their own copy of the FOPVAL file prior to commencing operation. To simplify the design for the ADM, especially if the FOPVAL parameters are to be monitored and modified by the maintenance engineer, it would be best to have only a single copy of each parameter in the software (rather than the 15 copies that exist presently).

Table 3.8-2 FilterInit OPVAL File Parameters - Role Applicability

5	Time Initialization	Y	Access to logging rate only required (param INIT_OutputData Rate)	N
1	Special Options Init	Y	Access to INIT_ProcessASF flag only	N
11	System State Standard Deviations	Y	Y - Position Error Only #	Y - Position Error Only #
62x2	Markov State Standard Deviations and Corr Times	Y	N	N
1	Q - Expand Limit (limit on statistical description of the unmodelled errors contained in the Kalman filter Q-Matrix)	Y	N	N
10	Q- Coeff Mx	Y	N	N
52	Initial Measurements	Y	Y	N
18	Lever Arm B-Vx (defines the positions of the various sensors used in the DIINS processing)	Y	Y	N

Notes:

- INIT_OutputDataRate: There is probably need to define OSL_Data (Task_Rate) and logging rate separately.

- # The ADM software will derive (a new algorithm is required here) the initial position error from the available information, so that this does not have to be entered by the operator.

3.8.4.2 Filter Configuration Files (e.g. S1V1P2L.cfg)

Table 3.8-3 indicates the parameters maintained in the filter configuration files.

Table 3.8-3 is based on the same configuration file as is presented in Appendix B, i.e. S1V1P2L.cfg. The number of parameters will vary depending on the particular parameter file considered.

The SYSTEM_STATES, MARKOV_STATES and MEASUREMENTS in **Table 3.8-3** just indicate (by indicating the name of the program constant) the parameters used from the FOPVAL file for the particular filter. Hence, the actual initial value of the parameters for the particular filter configurations are contained in the FOPVAL file.

Table 3.8-3 Filter Configuration File Parameter - Role Applicability

5	SENSORS used in configuration	Y	N	N
7	SYSTEM_STATES	Y	N	N
16	MARKOV_STATES	Y	N	N
10	MEASUREMENTS	Y	N	N

3.8.5 HCI Prototype Forms & Screens

3.8.5.1 DIINS ADM Initialization and Maintenance User Interface Forms

Appendix D indicates the outputs from the DIINS ADM HCI prototyping activities conducted during Phase 1.

3.8.5.2 SHINNADS Electronic Chart Display Screens

3.8.5.2.1 Overview

There are two DIINS SHINNADS display screen layouts:

- Basic (Figure 3.8.5-1) - This screen is designed to display the best solution position of the vessel while maximizing the chart display and supplying the user with other useful navigation information, mainly vector, route and cursor information. This was accomplished by displaying the best solution information in the top leftmost alpha window. The best solution alpha window contains the current DIINS latitude, longitude, heading, CMG, SMG and the estimated error (for 95% probability) along with the north and east standard deviations. Directly under this window is the DIINS status window that displays the devices currently being integrated. The remainder of the information is ECPINS standard navigation information.

- Detailed (Figure 3.8.5-2) - This screen is designed similar to the DIINS basic display except that it has a smaller chart window to provide an extra row of alpha windows for DIINS device information monitoring at the bottom of the screen. The bottom row consists of a Filter alpha window containing the current combination of devices being used for the filtered position. The remaining alpha windows contain the raw data output for each of five DIINS devices. The data displayed includes latitude, longitude, HDG, SMG, CMG and status.

3.8.5.2.2 DIINS Screen Layout

The DIINS screen layouts are found only on ECPINS® systems connected to the DIINS positioning system. The DIINS screen layouts display the resultant input from the DIINS system and from its individual sensors. These are:

- Inertial Navigation System 1 (INS1);
- Inertial Navigation System 2 (INS2);
- GPS1 (PPS);
- GPS2; and
- LORAN C.

There are two DIINS screens:

- DIINS Basic; and
- DIINS Detailed.

The difference between the two layouts is a row of text boxes along the bottom of the screen below the Chart window used to display raw sensor data from each of the DIINS Solution devices.

The common text boxes are arranged in a vertical column along the left side of the display as follows (top to bottom):

Text Box		Legend
DIINS Best Solution		
Lat:	Latitude	in Degrees, Decimal Minutes (ddd, mm.mmm)
Lon:	Longitude	in Degrees, Decimal Minutes (ddd, mm.mmm)
Heading:	Heading	in Decimal Degrees (ddd.d)
CMG:	Course Made Good	in Degrees Decimal Degrees (ddd.d)
SMG:	Speed Made Good	in Knots (kk.k)
Est. Error (95%)	Estimated Error	in Metres with 95% confidence level (mmm.m).
	N standard deviation	in Metres (mmm.m) (square root of covariance).
N StdDev:	E standard deviation	in Metres (mmm.m) (square root of covariance).
E StdDev:		

DIINS STATUS		
Status:	Status	OK or FAILED <device name>: based on the DIINS FDIR [Fault Detection, Isolation, and Reconfiguration] Status Bits abcdef where: a,b,c,d,e = 0 or 1; f = 1, 2, or 3
Currently Integrating:		Currently Integrating the list of devices that are currently being used to resolve the DIINS solution.

VECTORS		
Vectors:	Vectors	mode (STD or VAR) and look ahead time (MM:SS) of current vector used to display ships direction and speed.
EBL/VRM:	Electronic Bearing Line/ Variable Range Marker	the type of Electronic Navigation Aid currently used.
Bearing:	Bearing	the true angle that an Electronic Line creates from its point of origin.
Range:	Range	the distance a range marker is from its point of origin.

Sounding Units:	Sounding Units	the units used on the current chart for soundings. (IMO requirement).

ROUTE		
Route:	Route Name	
XTD:	Cross Track Distance	in Metres / Nautical Miles (mm.m / nm.nm) or other units (see Note above)
WP Bearing:	Way Point Bearing	bearing to the next WP in Decimal Degrees (ddd.d)
Leg Course:	Leg Course	in Decimal Degrees (ddd.d)
DST (SOR):	Distance	in Nautical or Statute Miles (to start of turn radius).
TTG (SOR):	Time To Go	in Hours, Minutes, Seconds (hh:mm:ss)
Dest ETA:	Destination Estimated Time of Arrival	in Hours, Minutes, Seconds (hh:mm:ss)
Next CTS:	Next Course to Steer	in Decimal Degrees (ddd.d).
Chart Offset:	Chart Offset	Radar Chart Offset indicator (IMO requirement).

CURSOR		
Cursor:	Cursor Terrain Feature	(Depth,...) Meter Range (mm-mm)
Bearing:	Bearing from Own Ship	in Decimal Degrees (ddd.d)
Range:	Range from Own Ship	in Metres and Nautical Miles (mmm.m, nm.nm) or other units (see Note above)
Lat:	Latitude	in Degrees, Decimal Minutes (ddd, mm.mmm)
Lon:	Longitude	in Degrees, Decimal Minutes (ddd, mm.mmm)

The DIINS Detailed screen contains a horizontal row of text boxes located directly below the chart display in. These are (from left to right):

GPS1		
Lat:	Latitude	in Degrees, Decimal Minutes (ddd, mm.mmm)
Lon:	Longitude	in Degrees, Decimal Minutes (ddd, mm.mmm)
HDG:	Heading	in Decimal Degrees (ddd.d)
SMG:	Speed Made Good	in Knots (kk.k)
CMG:	Course Made Good	in Degrees Decimal Degrees (ddd.d)
Status:	Status	OK or FAILED based on the DIINS FDIR Status Bits abcdef

GPS2		
Lat:	Latitude	in Degrees, Decimal Minutes (ddd, mm.mmm)
Lon:	Longitude	in Degrees, Decimal Minutes (ddd, mm.mmm)
HDG:	Heading	in Decimal Degrees (ddd.d)
SMG:	Speed Made Good	in Knots (kk.k)
CMG:	Course Made Good	in Degrees Decimal Degrees (ddd.d)
Status:	Status	OK or FAILED based on the DIINS FDIR Status Bits abcdef

INS1		
Lat:	Latitude	in Degrees, Decimal Minutes (ddd, mm.mmm)
Lon:	Longitude	in Degrees, Decimal Minutes (ddd, mm.mmm)
HDG:	Heading	in Decimal Degrees (ddd.d)
SMG:	Speed Made Good	in Knots (kk.k)
CMG:	Course Made Good	in Degrees, Decimal Degrees (ddd.d)
Status:	Status	OK or FAILED based on the DIINS FDIR Status Bits abcdef

INS2		
Lat:	Latitude	in Degrees, Decimal Minutes (ddd, mm.mmm)
Lon:	Longitude	in Degrees, Decimal Minutes (ddd, mm.mmm)
HDG:	Heading	in Decimal Degrees (ddd.d)
SMG:	Speed Made Good	in Knots (kk.k)
CMG:	Course Made Good	in Degrees, Decimal Degrees (ddd.d)
Status	Status	OK or FAILED based on the DIINS FDIR Status Bits abcdef

Loran		
Lat:	Latitude	in Degrees, Decimal Minutes (ddd, mm.mmm)
Lon:	Longitude	in Degrees, Decimal Minutes (ddd, mm.mmm)
HDG:	Heading	in Decimal Degrees (ddd.d)
SMG:	Speed Made Good	in Knots (kk.k)
CMG:	Course Made Good	in Degrees, Decimal Degrees (ddd.d)
Status:	Status	OK or FAILED based on the DIINS FDIR Status Bits abcdef

Below these text boxes run the Anti-Grounding and Most Recent Alarm Display. These two text boxes are common to all screen layouts.

NOTE

ECPINS® offers a choice of units. These come in three different sets:

(1) metres (m), nautical miles (nm), and knots (kt);

(2) feet (ft), nautical miles (nm), and knots (kt); and

(3) feet (ft), miles (mi), and miles per hour (mph).

The threshold value at which metres and feet become nautical or statute miles is 0.1 nm or 0.1 mi. The following give the value of transition:

(1) 160.9 m is displayed as 0.1 mi.

(2) 527.9 ft is displayed as 0.1 mi.

(3) 608 ft is displayed as 0.1 nm.

(4) 185.3 m is displayed as 0.1 nm.

Furthermore, decimal minutes can be displayed in their non-decimal form.

Figure 3.8.5-1 DIINS Basic Electronic Chart Display Screen

Figure 3.8.5-2 DIINS Detailed Electronic Chart Display Screen

3.9 Sensor Interfaces

As indicated in **Figure 3-1** most of the physical interfaces to the various navigation sensors are standard RS422 or RS232. In most cases the data format conforms to the standards of NMEA 0183, with additional sentences added to carry DIINS specific data where necessary. The requirement to make available the high rate (>=25 Hz) pitch, role and heading data will require an additional RS-422 interface to be provided to distribute the data.

The DIINS ADM configuration and interfaced sensors will be defined by parameters that include the following:

- Kalman filter error and measurement model parameters
- Kalman filter state and measurement selection specifications
- Kalman filter timing parameters (filter update rate, measurement construction rate)
- Serial digital communication parameters (e.g. baud rate, data bits, stop bits, parity for each RS-232 port)
- FDIR control parameters

The DIINS ADM will allow the key configuration parameters to be specified via a Graphical User Interface (GUI) during initialization, and for some of the parameters to be modified in real-time via the GUI where appropriate.

With the replacement of the existing CPU and memory cards with a single slot Single Board Computer such as the 68060, it becomes possible to output the high speed pitch, roll and heading data onto an Ethernet LAN if desired. This option may be preferable considering the architectures of today's modern Fire Control and Command and Control systems.

3.9.1 GPS/dGPS Interface

Among the possible navigation sensors that the DIINS ADM will likely need to work with, the GPS receiver will likely be the least expensive and, hence, the most likely sensor to be upgraded or replaced. It is, therefore, likely that the DIINS ADM will be required to interface to several different GPS receivers. It is reasonable to assume that most if not all candidate GPS receivers will implement some or all of the GPS messages defined in the NMEA 0183 standard Ref [8]. This report, therefore, recommends that the DIINS ADM interface to a generic dGPS-capable receiver should be a RS-232/422 serial interface that handles the GPS messages defined in the NMEA 0183 standard [8].

The minimum subset of defined NMEA messages required by the DIINS ADM comprises the GGA message. All GPS receivers that output NMEA messages will output the GGA message. It contains the following data items:

- UTC time of position fix;

- Geographic position coordinates;
- GPS quality factor (C/A or dGPS);
- Horizontal dilution of position (HDOP);
- Age of differential corrections; and
- Differential reference station identification.

If the DIINS ADM were to receive only the GGA message, then it would obtain the minimum information needed to construct Kalman filter GPS measurements and GPS error model parameters. The DIINS ADM would need to make certain assumptions about GPS position errors, based on the GPS quality factor, HDOP, the age of differential corrections and the reference station identification, for the purpose of constructing the Kalman filter GPS error and measurement models.

Other messages that contribute supplementary information include the GSA and GST messages. The GST message, for example, contains the following data items:

- UTC time of position fix;
- Corrected pseudorange error aggregate standard deviation or RMS;
- Error ellipse semi-major and semi-minor axes standard deviations and orientation; and
- Latitude, longitude and altitude error standard deviations.

These data can be used to compute an improved GPS error model, and to determine the GPS solution accuracy level. GPS performance at the C/A GPS, P/Y GPS or dGPS accuracy levels can be determined from the GST error ellipse parameters.

In such a generic GPS interface the DIINS software algorithm operation would depend on the particular GPS sensor used, and would configure the DIINS Kalman filter parameters according to the information available in the NMEA messages from the receiver. The detailed design of the DIINS-GPS interface will be performed during the detailed design phase of the ADM project, and will depend in part on the form of the dGPS error model also to be developed during the detailed design phase. One of the goals of the DIINS ADM development would be to implement a GPS receiver interface that is independent of the particular receiver being used.

3.10 System Improvements

A number of system improvements have been identified for the DIINS XDM in the RFP Statement of Work, Appendix C. These proposed improvements have been analyzed during the DIINS ADM system design review. Requirements for these improvements have been placed in Appendix A of this report, and the design features necessary to achieve these improvements have been included in the remainder of this report.

4. DIINS ADM SYSTEM DESIGN SUMMARY

This section briefly summarizes the key features of the proposed DIINS ADM system design.

General

- System requirements as per Appendix A
- For a diagram of the DIINS ADM system context see Figure 3-1. The basic software architecture will be as per Figure C-1 in Appendix C.

Operation

- The DIINS ADM basic operation will be as per Appendix G.

User Interface

- The User Interface will be as per Section 3.8.5.2 and Appendix D, with the interface between the SHINNADS and DIINS being as per the NMEA sentences defined in Appendix H.

Configuration Parameters

- The DIINS configuration parameters will be as per Appendix I, with the manipulation of the parameters as per Appendix G.

Data Logging

- As per Appendix E

Filter Configuration

- The DIINS filter configuration will vary depending on the attached sensors. The main filter configurations to be tested during development will be the 11 and 15 filter configurations defined in Sections 3.2.3.3 and 3.2.3.6 .

High Rate Attitude Data Distribution

- Will be via an RS422 interface, with the data being distributed at >= 25Hz.

Hardware Configuration

- The DIINS ADM will consist of the following circuit boards mounted in the SHINNADS supplied VME slots:

  - MVME-177-004 - 68060 50 MHz, 32 MB, 2 MB PROM processor board
  - Force SYS68K ISIO-2 - 8 Port RS232/RS422 Intelligent Serial Interface
  - Force SYS68K ISIO-1FP - Serial interface Front Panel Transition Module

- Approximate hardware costs per ADM unit will be $16,155, as per Table 3.6-2, Option 5.

- For a cost summary discussion, see Section 3.6.4.

Software Configuration

- Ada 83 software developed on the existing DIINS XDM development environment.

- For the proposed software architecture updates, see Appendix C.

Alternative Implementation Option

- If funding were made available for the Development Phase, consideration could be given to implementing the DIINS ADM using more modern technology, e.g. VME Pentium processor, GreenHills Ada 95 compiler and VxWorks run-time environment. The main drawback with this approach, apart from the lack of funding for the SDE, is the increased risk, in terms of cost and schedule, of porting the software to a new processor and compiler. These risks could be mitigated to some extent by the use of the modern software development environment that this option provides. See Sections 3.6.2 and 3.6.4 for the details of this option.

5. LIST OF ACRONYMS

ADM	Advanced Development Model
CISC	Complex Instruction Set Computer
dGPS	Differential Global Positioning System
DIINS	Dual Inertial Integrated Navigation System
DND	Canadian Department of National Defence
DoD	US Department Of Defense
DREO	Defence Research Establishment Ottawa
ECDIS	Electronic Chart and Display Information System
FDIR	Failure Detection, Isolation and Reconfiguration
GFE	Government Furnished Equipment
GPS	Global Positioning System
GUI	Graphical User Interface
INS	Inertial Navigation System
MIPS	Millions of Instructions Per Second
MTTR	Mean Time To Repair
NMEA	National Marine Electronics Association
OSL	Offshore Systems Ltd.
RFP	Request For Proposal
SA	Selective Availability
SDE	Software Development Environment
SHINNADS	SHipboard INtegrated NAvigation and Display System
SINS	Strapdown Inertial Navigation System
XDM	eXperimental Development Model

APPENDIX A REQUIREMENT FOR DIINS

GENERAL

1. This document details the requirement for the Dual Inertial Integrated Navigation System. (DIINS) which will be installed in major Canadian naval vessels (initially Halifax class) fitted with dual, high accuracy marine inertial navigation systems (Mk29, Mk49, WSN-5). DIINS must operate worldwide in all areas of operations of the Canadian Navy. The system will be capable of providing vessel position, speed, heading, and attitude data to SHINNADS (the Shipboard Integrated Navigation and Display System that is being built for the navy by Offshore Systems Ltd of Vancouver B.C.) and distribute the same to other users on the ship. DIINS will incorporate sophisticated Kalman filtering techniques to estimate and reduce the errors in the inertial systems by using the position fixes available from the GPS (SPS, PPS, and/or DGPS) and other radio receivers (Loran) on board. DIINS will be capable of detecting and isolating faults in the navigation sensors and will automatically reconfigure its integration filters to remove failed or questionable sensors from the navigation solution.

SYSTEM FUNCTIONAL REQUIREMENTS

2. System Configuration

A	The DIINS system shall be a subsystem of SHINNADS.	A.2	Hardware (H)
B	DIINS will reside within the SHINNADS console.	A.2	H
C	Communication between the DIINS processor and the SHINNADS processor will be via a link internal to the console.	A.2	H & Software (S)
D	In the ADM, this link shall be a two way RS-422 link to allow for DGPS data to be provided to DIINS from SHINNADS, and in addition, to allow for complete control and display of the DIINS internal parameters and sensor and filter configuration.	A.2	H & S

3. System Components. Each of the components comprising the DIINS system shall be capable of meeting the following requirements.

Note: In the following table:

- "Parent Rqmt" is a reference back to the original requirement in the DIINS RFP, to support traceability of the requirements to Appendices A and C of the RFP. The numbers shown indicate the appropriate section within the indicated appendix.
- "Component" will eventually (in the final version of the Phase 1 report) indicate the system hardware or software components used to implement the requirement in the proposed system design.

A	*Processing Component.* This unit shall consist of Computer Card Assemblies (CCA's) compatible with the SHINNADS card cage.	A.3.1	H
B	The components chosen shall have sufficient processing capacity to run all the DIINS software under a maximal sensor configuration and at least TBD(15) parallel Kalman filters.	A.3.1	H
C	*Sensor Interface Component.* This unit shall consist of high speed CCA's capable of reading both RS232 and RS422 serial data from the various sensors.	A.3.2	H & S
D	*Control and Display Component.* This shall consist of two equivalent methods of accessing the DIINS internal parameters and configuration files: a) via the SHINNADS console and the two way serial communication link to DIINS b) via a separate hand-held or laptop device attached to an available communications port on DIINS.	A.3.3	H & S
E	*Data Distribution Component:* This unit shall consist of CCA's capable of distributing the output data at the rates specified in Section 7 of the requirements.	A.3.4	H & S
F	*Data Recording Component.*: This unit shall be capable of logging DIINS navigation outputs and other, user-selectable Kalman filter outputs, at rates and durations specified by an operatcr using either of the control and display methods above.	A.3.5	H & S

	The data to be logged shall be as follows: Raw Sensor Data: • INS • GPS & DGPS • Loran Kalman Filter Outputs: • Filter Residuals • State Vector • Position Estimate • Covariances Miscellaneous • Identified software exceptions • FDI Log text messages • Filter Log text messages		
G	The maximum volume of data that can be logged without changing the removable hardware media shall be: • 1GB on Portable PC harddrive • TBD (4 or 8GB) on portable EXABYTE tape drive.	Derived from A.3.5	H & S
H	The data shall be logged with the format defined in Appendix E (compression may be used).	Derived from A.3.5	H & S
I	The DIINS data shall be logged in a sequence of log files, each file covering a two hour time period, with no data gaps between files.	Derived from A.3.5	S
J	The names of the log files shall be automatically generated by the system, and sill be based on a function of DIINS time.	Derived from A.3.5	S
K	The logged data shall be able to be replayed at an on-shore analysis facility.	Derived from A.3.5	H & S

4. Sensor Requirements

A	The system shall have the ability to interface to at least the following sensors: Either: - Dual Sperry Mk-29 Mod 3 Inertial Navigation Systems or - (pending DND procurement approval for the MK-49) - Dual Sperry Mk-49 (NATO SINS) Inertial Navigation System and - Trimble TASMAN (P/Y) GPS receiver. - Leica 12 channel MX94xx series DGPS receivers *Note: The Leica DGPS is located in the SHINNADS system.* - Internav GC500 combined Loran C / GPS(C/A) receiver Or the - Internav LC360 Loran C receiver *Note: It is assumed that DIINS sees the GC500 only as a Loran C system, with outputs identical to the LC360.* *Note: The Internav GC500 is replacing the Internav LC360 Loran-C system in service. DIINS will have to still maintain the capability to process the LC360 in the foreseeable future.* - Differential GPS (DGPS), from a system external to SHINNADS	A.4 C.3	H & S
B	1. The allowable DGPS inputs shall be as follows: a. via a modified link with OSL's SHINNADS to allow the DGPS receiver within SHINNADS to provide	C.3	H & S

	DGPS data to DIINS; b. via a separate DGPS system that can supply DGPS data in NMEA formats, through an unused serial port on the DIINS ADM.		
C	The hardware interfaces shall be sufficiently general to allow the above sensors to be used in DIINS.	A.4	H
D	The DIINS software, in terms of simple data stream formats, Kalman filter error modelling and Fault Detection, and at the control and display level, must be able to accommodate at least the above sensors.	A.4, C.3	S
E	The DIINS ADM shall contain a parameterized sensor library, in which a few of the most common and likely naval INS (specifically the Sperry MK-49 and Litton WSN-5 (the INS used on the TRUMP Class ships)), GPS and other navigation systems can be selected from a list.	C.5	S
F	Minimal software changes shall be required to interface to a new sensor. Thus the areas of the code that must be changed to accommodate a new sensor must be kept as separate as possible and documented thoroughly.	C.5	S

5. System Accuracy

A	The system shall be capable of providing navigational accuracies equivalent to those obtained by the XDM, approximately 15-20m (95%) N and E position error and 1-2 arcmin (rms) roll pitch and heading error (Mk-29 integrated with Tasman P/Y GPS).	A.5	S

6. Fault Detection and Sensor Redundancy

A	The system shall implement FDIR algorithms equivalent to the Kalman filter residual testing, chi-squared fault detection and multi-filter failure isolation techniques within the DIINS XDM.	A.6	S
B	In addition the methodology, as presented in the XDM, of filter selection and reconfiguration under sensor failure conditions, shall be followed.	A.6	S

7. Data Distribution Requirements

A	The best integrated estimate of vessel position, speed and heading shall be output on an RS-422 serial line to SHINNADS at least once every 1.28 seconds.	A.7	H & S
B	The best estimate of vessel roll, pitch and heading shall be output at least every 40 milliseconds.	A.7	H & S
C	The contractor shall propose and implement a method for distributing this high rate roll, pitch and heading data (as well as the lower rate position and speed) from DIINS to the rest of the ship. There are 3 distribution requirements: a) Pos/speed/heading and internal DIINS diagnostic info to SHINNADS at ~ 1/second; (Currently met by XDM) b) Pos/speed/heading at ~1/sec for general shipboard system uses; and (Currently met by XDM) c) High rate (25 Hz or greater), low latency (less than or equal to 40 ms) Roll/Pitch/Heading data for weapons or other systems requiring precise, near-continuous attitude information. (New for ADM) *Note: The requirement is to distribute the high*	A.7 C.4	H & S

	rate data at a minimum of 25Hz. The design goal is to distribute the attitude data at a higher rate if possible, e.g. greater than or equal to 100Hz. The high rate data will be achieved by the redistribution of the data received from the INS, but with interpolation and corrections being applied within the ADM based on the DIINS filter outputs.		

8. Host Vessel

A	The host vessel will be a Halifax class frigate of the Canadian Navy.	A.8	H

9. Environmental Conditions

A	The DIINS system will be housed within the SHINNADS console on the bridge of the vessel.	A.9	H

10. Reliability

A	The system shall be designed according to generally accepted principles of ruggedized commercial computing systems. No mil-spec requirements are placed on DIINS.	A.10	H
B	The system reliability design target is a minimum of 5000 hours MTBF.	A.10	H

11. Maintainability.

A	The system shall be designed to minimize Mean time to Repair (MTTR).	A.11	H
B	Design objectives shall be 15 minutes for first-line (on vessel) repairs. Board level replacements are desirable on the vessel.	A.11	H

12. Human Factors Engineering

A	The system shall be designed to simplify and render most efficient the operator interface.	A.12	H & S
B	The DIINS ADM user shall be able to view and optionally change some of the DIINS configuration and filter parameters that are presently set in the DIINS configuration and filter parameter files (SimSetup.cfg and FilterInitx.dat). The parameter values currently in the DIINS XDM parameters files that are not defined below will be setup to appropriate values by the system during initialization. The parameters accessible to the ADM user will be as follows: Configuration Parameters: • Definition of the installed sensor models in DIINS; • Selection of the sensors to be used with DIINS; • Lever Arms for the installed sensors; • RS-232/422 communication channel parameters for the installed sensors; Initial sensor measurement values (for filter initialization), including standard deviations and failure and pass fractions and times.	C.2	S

C	There shall be two essentially equivalent ways to view and change parameters: a. via a hand held of laptop computer (external control and display unit) interfaced to DIINS through an unused serial port; and b. via a custom screen on the SHINNADS console.	C.2	H & S
D	The control and display functions for the DIINS internal parameters, when presented on the SHINNADS console screen shall be consistent with other control and display screens used in SHINNADS.	A.12	S
E	The external control and display unit shall present the similar interface as used when accessing DIINS from the SHINNADS console.	A.12	S
F	The operator shall be able to specify the source (if any) of DGPS data as one of the following two sources: a. SHINNADS DGPS Receiver b. Separate DGPS system	C.3	S
G	The operator shall be able to select the INS , GPS and other navigation sensors/systems to be used from a list of the available sensors/systems.	C.5	S
H	The sensor and Kalman filter configuration interface shall be sufficiently flexible to allow a knowledgeable user the ability to fine tune the parameters of a particular sensor.	C.5	S
I	The DIINS ADM (via the SHINNADS display) shall provide the following outputs when operating in the Maintenance Mode: • The Percentage used of the available processing power • The filter currently selected for output • Identified software exceptions • FDI Log text messages • Filter Log text messages	Data previously available via development lab display.	S

J	When operating in Maintenance Mode the DIINS software shall allow the operator to inject a hard or soft sensor failure for a selected sensor, which will cause the sensor data to be degraded accordingly. This will be used to verify the correct operation (including failure detection performance) of the DIINS failure detection processing. *Note: This feature can also be used to support acceptance testing of this functionality.*	Derived from A.6	S
K	The DIINS software shall be able to be started without the need to use the HCI to configure the system.	Derived from A.2	
L	Once the system has been installed and configured, this configuration shall be used as the default configuration.	Derived from A.2	
M	The operator shall have to take some specific action to allow the configuration control HCI forms to be presented.	Derived from A.2	

APPENDIX B DIINS CONFIGURATION FILES

This appendix indicates the format of the configuration files used in the DIINS XDM. For details on the proposed approach to handling configuration data in the ADM see Appendix I.

Note: The details of the actual parameter values have been removed from this appendix, and have been replaced by XXXXX's.

----------[Content-type: text/plain; name=SimSetup.README]----------

SimSetup.cfg Description for x Filters

Line 1

REALTIME/SIMULATION

.REALTIME - Run forever

.SIMULATION - Run for a certain amount of time and usually from previously collected data. Also logs some messages to disk.

Line 2

BIG_FILE/SEPARATE_FILES

.DataLogging using BIG_FILE goes to TAPE or DISK as a single file.

.DataLogging using SEPARATE_FILES goes ONLY to DISK.

Lines 3-19

Are the names of all possible sensor models in "AllSensorModels" (SimConfig.spc) and their communication setup.

e.g. Line 3

Sensor SPERRY_MK29_INS1 come in at 1.28 secs from Port 1 using these comm. parmaeters (BAUD_9600,BITS_8,PARITY_NONE,STOP_1, HANDSHAKE_NONE)

Note: Port 0 means the sensor is not used.

Line 20-25

Are the names of all possible non-sensor models in "AllNonSensorTasks" (SimConfig.spc) and their communication setup.

This is the same format to above.

Note 1: Port has meaning for only OSL_TASK
Note 2: PREFILTER_TASK does not use data rate (1.28), this is
reset to PhiRate.
Note 3: The FDI_TASK, ASF_TASK, PPC_TASK, AO_TASK use data
rates, but no ports.

Line 26
Number of Filters - 15 or less

Line 27
Simulation Time To Run - Used only in SIMULATION

Line 28-42 (changes with "Number of Filters")
Simulation configuration for each filter
(see Simulation.README)

Line 43-59
If SIMULATION is used, these files will be opened and read
as sensor data (strict format)

Line 60-74 (changes with "Numbers of Filters")
Filter Initialization file for each filter
(see FOPVAL.README)

Line 75-89
Filter output files - Only used if SEPARATE_FILES (line 2) are
used.

e.g. Line 75

NFS_RES:Disp_LT97_1.bRes NFS_RES:State_LT97_1.bRes
NFS_RES:Res_LT97_1.bRes
Each filter has 3 output files (Display, States & Residuals).

Line 90-104
Filter logging files - ASCII messages from each Filter

Line 105
FDI logging file - ASCII message from the FDI task

```
----------[ Content-type: text/plain; name=SimSetup.cfg ]----------
REALTIME
BIG_FILE
SPERRY_MK29_INS1          1.28   1     BAUD_9600   BITS_8   PARITY_NONE
STOP_1 HANDSHAKE_NONE
LITTON_LTN90_INS1         1.0          0
```

```
SIM_INS1                    1.0          0
SPERRY_MK29_INS2            1.28   3     BAUD_9600   BITS_8   PARITY_NONE
STOP_1 HANDSHAKE_NONE
LITTON_LTN90_INS2           1.0          0
SIM_INS2                    1.0          0
ASHTECH_XII_GPS             1.0          0
TRIMBLE_TANS_PY_GPS1        1.0          6     BAUD_9600          BITS_8
PARITY_NONE STOP_1 HANDSHAKE_NONE DontUseMyClock
TRIMBLE_TANS_PY_GPS2        1.0          7     BAUD_9600          BITS_8
PARITY_NONE STOP_1 HANDSHAKE_NONE UseMyClock
UofC_dGPS1                  1.0          0
UofC_dGPS2                  1.0          0
TRIMBLE_VECTOR_GPS          1.0          0
SIM_GPS                     5.0          0
SIM_OMEGA                   10.0   0
INTERNAV_LC360_LORAN        1.0    5     BAUD_2400   BITS_8   PARITY_NONE
STOP_1 HANDSHAKE_NONE
SIM_LORAN                   10.0   0
SIM_SPEED_LOG               1.0          0
PREFILTER_TASK              1.28   0     -- This is reset to PhiRate
FDI_TASK                    1.28   0
ASF_TASK                    300.0  0
PPC_TASK                    30.0   0
AO_TASK                     60.0   0
OSL_TASK                    0.0          8     BAUD_19200         BITS_8
PARITY_NONE STOP_1 HANDSHAKE_NONE
15
200000.0
REX$RAM:S12.cfg
REX$RAM:S1V1P2L.cfg
REX$RAM:S2V1P2L.cfg
REX$RAM:S1V1P2.cfg
REX$RAM:S2V1P2.cfg
REX$RAM:S1V1L.cfg
REX$RAM:S2V1L.cfg
REX$RAM:S1P2L.cfg
REX$RAM:S2P2L.cfg
REX$RAM:S1V1.cfg
REX$RAM:S2V1.cfg
REX$RAM:S1P2.cfg
REX$RAM:S2P2.cfg
REX$RAM:S1L.cfg
REX$RAM:S2L.cfg
temp_a.bin temp_v.bin temp_p.bin -- SPERRY_MK29_INS1
temp_a.bin temp_v.bin temp_p.bin -- LITTON_LTN90_INS1
```

temp_a.bin temp_v.bin temp_p.bin -- SIM_INS1
temp_a.bin temp_v.bin temp_p.bin -- SPERRY_MK29_INS2
temp_a.bin temp_v.bin temp_p.bin -- LITTON_LTN90_INS2
temp_a.bin temp_v.bin temp_p.bin -- SIM_INS2
temp.bin -- ASHTECH_XII_GPS
temp.bin -- TRIMBLE_TANS_PY_GPS1
temp.bin -- TRIMBLE_TANS_PY_GPS2
temp.bin -- UofC_dGPS1
temp.bin -- UofC_dGPS2
temp.bin -- TRIMBLE_VECTOR_GPS
temp.bin -- SIM_GPS
temp.bin -- SIM_OMEGA
temp.bin -- INTERNAV_LC360_LORAN
temp.stn temp.bin -- SIM_LORAN
temp.bin -- SIM_SPEED_LOG
REX$RAM:FOPVAL1.dat
REX$RAM:FOPVAL2.dat
REX$RAM:FOPVAL3.dat
REX$RAM:FOPVAL4.dat
REX$RAM:FOPVAL5.dat
REX$RAM:FOPVAL6.dat
REX$RAM:FOPVAL7.dat
REX$RAM:FOPVAL8.dat
REX$RAM:FOPVAL9.dat
REX$RAM:FOPVAL10.dat
REX$RAM:FOPVAL11.dat
REX$RAM:FOPVAL12.dat
REX$RAM:FOPVAL13.dat
REX$RAM:FOPVAL14.dat
REX$RAM:FOPVAL15.dat
NFS_RES:Disp_LT97_1.bRes NFS_RES:State_LT97_1.bRes NFS_RES:Res_LT97_1.bRes
NFS_RES:Disp_LT97_2.bRes NFS_RES:State_LT97_2.bRes NFS_RES:Res_LT97_2.bRes
NFS_RES:Disp_LT97_3.bRes NFS_RES:State_LT97_3.bRes NFS_RES:Res_LT97_3.bRes
NFS_RES:Disp_LT97_4.bRes NFS_RES:State_LT97_4.bRes NFS_RES:Res_LT97_4.bRes
NFS_RES:Disp_LT97_5.bRes NFS_RES:State_LT97_5.bRes NFS_RES:Res_LT97_5.bRes
NFS_RES:Disp_LT97_6.bRes NFS_RES:State_LT97_6.bRes NFS_RES:Res_LT97_6.bRes
NFS_RES:Disp_LT97_7.bRes NFS_RES:State_LT97_7.bRes NFS_RES:Res_LT97_7.bRes
NFS_RES:Disp_LT97_8.bRes NFS_RES:State_LT97_8.bRes NFS_RES:Res_LT97_8.bRes
NFS_RES:Disp_LT97_9.bRes NFS_RES:State_LT97_9.bRes NFS_RES:Res_LT97_9.bRes
NFS_RES:Disp_LT97_10.bRes NFS_RES:State_LT97_10.bRes NFS_RES:Res_LT97_10.bRes
NFS_RES:Disp_LT97_11.bRes NFS_RES:State_LT97_11.bRes NFS_RES:Res_LT97_11.bRes
NFS_RES:Disp_LT97_12.bRes NFS_RES:State_LT97_12.bRes NFS_RES:Res_LT97_12.bRes
NFS_RES:Disp_LT97_13.bRes NFS_RES:State_LT97_13.bRes NFS_RES:Res_LT97_13.bRes
NFS_RES:Disp_LT97_14.bRes NFS_RES:State_LT97_14.bRes NFS_RES:Res_LT97_14.bRes
NFS_RES:Disp_LT97_15.bRes NFS_RES:State_LT97_15.bRes NFS_RES:Res_LT97_15.bRes

NFS_RES:Setup_LT97_1.cfg
NFS_RES:Setup_LT97_2.cfg
NFS_RES:Setup_LT97_3.cfg
NFS_RES:Setup_LT97_4.cfg
NFS_RES:Setup_LT97_5.cfg
NFS_RES:Setup_LT97_6.cfg
NFS_RES:Setup_LT97_7.cfg
NFS_RES:Setup_LT97_8.cfg
NFS_RES:Setup_LT97_9.cfg
NFS_RES:Setup_LT97_10.cfg
NFS_RES:Setup_LT97_11.cfg
NFS_RES:Setup_LT97_12.cfg
NFS_RES:Setup_LT97_13.cfg
NFS_RES:Setup_LT97_14.cfg
NFS_RES:Setup_LT97_15.cfg
NFS_RES:FDI.log

----------[Content-type: text/plain; name=FOPVAL.README]----------
Description of FOPVALx.dat

Line 2-3 INIT_PhiRate
Initialize PhiRate (per PhiPeriod)

Line 5-6 INIT_PropagateRate
Initialize PropagateRate (per PhiPeriod)

Line 8-9 INIT_UpdateRate
Initialize UpdateRate (per PhiPeriod)

Line 11-12 INIT_OutputDataRate
Initialize OutputDataRate (per PhiPeriod)

Line 14-18 INIT_ProcessChiSquared
Initalize Chi Squared processing parameters

.TRUE/FALSE - ChiSquared Activated
.Levelxxx - ConfidenceLevelType (ChiSquaredFailureDetection.pkg)
.Chi-Squared Process period (per UpdatePeriod)
.Failure Trigger Time (value + unit)

e.g.
INIT_ProcessChiSquared

TRUE -- ChiSquared.Activated ??
Level99 -- ChiSquared.ConfidenceLevel
1 -- UpdatePeriodsPerChiSqProcessPeriod
2.0 minutes -- ChiSquared.FailureTriggerTime

Line 21-22 INIT_ProcessAO
Process Auto Omega?

Line 24-25 INIT_ProcessPPC
Process PPC?

Line 27-28 INIT_ProcessASF
Process ASF?

Line 32-44 INIT_SystemStates
Initialize system states is divided into:
. Name of system state, AllSystemStates (SimTypes.spc)
. Standard Deviation and unit

Line 48-111 INIT_MarkovStates
Initialize markov states is divided into:
. Name of Markov state, AllMarkovStates (SimTypes.spc)
. Standard Deviation and unit
. Correlation time and unit

Line 11-115 INIT_Q_ExpandLimit
Series expansion limit

Line 118-129 INIT_Q_CoeffVx
Model of Spectral density matrix of the random forcing functions:

1) correlation distance of horizontal gravity anomalies
2) correlation distance of vertical gravity anomalies
3) standard deviation of horizontal gravity anomalies
4) standard deviation of vertical gravity anomalies
5) correlation time of random accelerometer scale factors
6) standard deviation of random accelerometer scale factors
7) standard deviation of the accelerometer misalignments
8) correlation time for the gyro mass unbalance error
9) standard deviation for the gyro mass unbalance error
10)spectral density of stationary gyro drift error

Line 134-187 INIT_Meas
Initialize measurement is divided into:

. Name of measurement, AllMeas (SimTypes.spc)
. Standard deviation of measurement noise (sigma) and unit
. Residual test limit (N-sigma)
. Fraction of residual tests failing in last period of time
(Frac & Time)
. Fraction of residual tests passing in last period of time
(Frac & Time)
. unit of Fractions (above)

Line 191-209 INIT_LeverArmB_Vx
Initialize Lever-Arm is divided into:
. Name of Sensor Model, AllSensorModels (SimConfig.spc)
. x, y, z offset
. unit of offsets

Note: ALL sensors are listed, but only those sensors
enabled are used.

----------[Content-type: text/plain; name=FOPVALx.dat]----------
---------- Time Init
INIT_PhiRate
1 -- INS_DataPeriodsPerPhiPeriod

INIT_PropagateRate
1 -- PhiPeriodsPerPropagatePeriod

INIT_UpdateRate
7 -- PropPeriodsPerUpdatePeriod (Last=7)

INIT_OutputDataRate
7 -- INS_DataPeriodsPerOutputPeriod (Last=14)

INIT_ProcessChiSquared
TRUE -- ChiSquared.Activated ??
Level99 -- ChiSquared.ConfidenceLevel
1 -- UpdatePeriodsPerChiSqProcessPeriod
2.0 minutes -- ChiSquared.FailureTriggerTime (s)

---------- Special Options Init
INIT_ProcessAO
FALSE

INIT_ProcessPPC
FALSE

```
INIT_ProcessASF
        TRUE

-- SystemState      Std.Dev.  Units
-- -----------      --------  -----
INIT_SystemStates

        X_POS_ERR               XXXXXX    metres
        Y_POS_ERR               XXXXXX    metres
        Z_POS_ERR               XXXXXX    metres
        X_VEL_ERR               XXX       m_per_s
        Y_VEL_ERR               XXX       m_per_s
        Z_VEL_ERR               XXX       m_per_s
        X_ATT_ERR               XXXX      degrees
        Y_ATT_ERR               XXXX      degrees
        Z_ATT_ERR               XXXX      degrees
        GPS1_CLOCK_BIAS         XXXXX     metres
        GPS2_CLOCK_BIAS         XXXXX     metres

-- MarkovState      Std.Dev.  Units     CorrTime  Units
-- -----------      --------  -----     --------  -----
INIT_MarkovStates

        X_ACCEL_BIAS_ERR        XXXXX     micro_gs  XXXXX     seconds
        Y_ACCEL_BIAS_ERR        XXXXX     micro_gs  XXXXX     seconds
        Z_ACCEL_BIAS_ERR        XXXXX     micro_gs  XXXXX     seconds
        X_GYRO_BIAS_ERR         XXXXX     deg_per_hr XXXXX    hours
        Y_GYRO_BIAS_ERR         XXXXX     deg_per_hr XXXXX    hours
        Z_GYRO_BIAS_ERR         XXXXX     deg_per_hr XXXXX    hours
        X_LOG_VEL_ERR           XXX       m_per_s   XXXXXX    seconds
        Y_LOG_VEL_ERR           XXX       m_per_s   XXXXXX    seconds
        X_OCEAN_CURRENT_ERR XXX           m_per_s   XXXXXX    seconds
        Y_OCEAN_CURRENT_ERR XXX           m_per_s   XXXXXX    seconds
        X_GPS1_POS_ERR          XXXX      metres    XXXXX     seconds
        Y_GPS1_POS_ERR          XXXX      metres    XXXXX     seconds
        Z_GPS1_POS_ERR          XXXX      metres    XXXXX     seconds
        X_GPS1_VEL_ERR          XXX       m_per_s   XXXX      seconds
        Y_GPS1_VEL_ERR          XXX       m_per_s   XXXX      seconds
        Z_GPS1_VEL_ERR          XXX       m_per_s   XXXX      seconds
        X_GPS2_POS_ERR          XXXX      metres    XXXXX     seconds
        Y_GPS2_POS_ERR          XXXX      metres    XXXXX     seconds
        Z_GPS2_POS_ERR          XXXX      metres    XXXXX     seconds
        X_GPS2_VEL_ERR          XXX       m_per_s   XXXX      seconds
        Y_GPS2_VEL_ERR          XXX       m_per_s   XXXX      seconds
```

Z_GPS2_VEL_ERR	XXX	m_per_s	XXXX	seconds
GPS1_TIME_LAG	XXXX	seconds	XXXXXX	seconds
GPS2_TIME_LAG	XXXX	seconds	XXXXXX	seconds
GPS1_CLOCK_DRIFT	XXXX	m_per_s	XXXXX	seconds
GPS2_CLOCK_DRIFT	XXXX	m_per_s	XXXXX	seconds
GPS1_ROLL_ERR	XXX	arcmin	XXXXX	seconds
GPS1_PITCH_ERR	XXX	arcmin	XXXXX	seconds
GPS1_HDG_ERR	XXX	arcmin	XXXXXX	seconds
GPS1_ROLL_BIAS	XXXX	arcmin	XXXXX	seconds
GPS1_PITCH_BIAS	XXXX	arcmin	XXXXX	seconds
GPS1_HDG_BIAS	XXX	arcmin	XXXXX	seconds
GPS2_ROLL_ERR	XXX	arcmin	XXXXX	seconds
GPS2_PITCH_ERR	XXX	arcmin	XXXXX	seconds
GPS2_HDG_ERR	XXX	arcmin	XXXXXX	seconds
GPS2_ROLL_BIAS	XXXX	arcmin	XXXXX	seconds
GPS2_PITCH_BIAS	XXXX	arcmin	XXXXX	seconds
GPS2_HDG_BIAS	XXX	arcmin	XXXXX	seconds
GPS_PSR1_ERR	XXX	metres	XXXXXX	seconds
GPS_PSR2_ERR	XXX	metres	XXXXXX	seconds
GPS_PSR3_ERR	XXX	metres	XXXXXX	seconds
GPS_PSR4_ERR	XXX	metres	XXXXXX	seconds
GPS_PSR5_ERR	XXX	metres	XXXXXX	seconds
GPS_PSR6_ERR	XXX	metres	XXXXXX	seconds
GPS_PSR7_ERR	XXX	metres	XXXXXX	seconds
GPS_PSR8_ERR	XXX	metres	XXXXXX	seconds
GPS_PSR9_ERR	XXX	metres	XXXXXX	seconds
GPS_PSR10_ERR	XXX	metres	XXXXX0	seconds
GPS_PSR11_ERR	XXX	metres	XXXXXX	seconds
GPS_PSR12_ERR	XXX	metres	XXXXXX	seconds
LORAN_TDW_ERR	XXXXX	metres	XXXXXX	seconds
LORAN_TDX_ERR	XXXXX	metres	XXXXXX	seconds
LORAN_TDY_ERR	XXXXX	metres	XXXXXX	seconds
LORAN_TDZ_ERR	XXXXX	metres	XXXXXX	seconds
OMEGA_PHASE1_ERR	XXXXXX	metres	XXXXXXX	seconds
OMEGA_PHASE2_ERR	XXXXXX	metres	XXXXXXX	seconds
OMEGA_PHASE3_ERR	XXXXXX	metres	XXXXXXX	seconds
OMEGA_PHASE4_ERR	XXXXXX	metres	XXXXXXX	seconds
OMEGA_PHASE5_ERR	XXXXXX	metres	XXXXXXX	seconds
OMEGA_PHASE6_ERR	XXXXXX	metres	XXXXXXX	seconds
OMEGA_PHASE7_ERR	XXXXXX	metres	XXXXXXX	seconds
OMEGA_PHASE8_ERR	XXXXXX	metres	XXXXXXX	seconds

---------- Q_ExpandLimit
INIT_Q_ExpandLimit
6

```
------------------------------ Q_CoeffMx
INIT_Q_CoeffVx

        XXXX        kilometres
        XXXX        kilometres
        XXXX        micro_gs
        XXXX        micro_gs
        XXXX        seconds
        XXXX        ppm
        XXXX        arcsec
        XXXX        seconds
        XXXX        deg_per_hr_per_g
        XXXX        deg_per_SqrtHr

-- Measurement   Sigma Sigma Resid   Fail    Pass  Time
--                Units Test  Frac Time Frac Time Units
-- -----------   ----- ----- ------ ---- ---- ---- ---- -----
INIT_Meas

        X_LOG_VEL_MEAS        XXXX m_per_s XXXX XXXX XXX  XXXX XXX
                                                              minutes
        Y_LOG_VEL_MEAS        XXXX m_per_s XXXX XXXX XXX  XXXX XXX
                                                              minutes
        X_GPS1_POS_MEAS XXXX metres  XXXX XXXX XXX  XXXX XXX  minutes
        Y_GPS1_POS_MEAS XXXX metres  XXXX XXXX XXX  XXXX XXX  minutes
        Z_GPS1_POS_MEAS XXXX metres  XXXX XXXX XXX  XXXX XXX  minutes
        X_GPS1_VEL_MEAS       XXXX m_per_s XXXX XXXX XXX  XXXX XXX
                                                              minutes
        Y_GPS1_VEL_MEAS       XXXX m_per_s XXXX XXXX XXX  XXXX XXX
                                                              minutes
        Z_GPS1_VEL_MEAS       XXXX m_per_s XXXX XXXX XXX  XXXX XXX
                                                              minutes
        X_GPS2_POS_MEAS XXXX metres  XXXX XXXX XXX  XXXX XXX  minutes
        Y_GPS2_POS_MEAS XXXX metres  XXXX XXXX XXX  XXXX XXX  minutes
        Z_GPS2_POS_MEAS XXXX metres  XXXX XXXX XXX  XXXX XXX  minutes
        X_GPS2_VEL_MEAS       XXXX m_per_s XXXX XXXX XXX  XXXX XXX
                                                              minutes
        Y_GPS2_VEL_MEAS       XXXX m_per_s XXXX XXXX XXX  XXXX XXX
                                                              minutes
        Z_GPS2_VEL_MEAS       XXXX m_per_s XXXX XXXX XXX  XXXX XXX
                                                              minutes
        GPS1_ROLL_MEAS XXXX arcmin  XXXX XXXX XXX  XXXX XXX  minutes
        GPS1_PITCH_MEAS       XXXX arcmin  XXXX XXXX XXX  XXXX XXX
                                                              minutes
```

```
GPS1_HDG_MEAS  XXXX arcmin XXXX XXXX XXX  XXXX XXX  minutes
GPS2_ROLL_MEAS XXXX arcmin XXXX XXXX XXX  XXXX XXX  minutes
GPS2_PITCH_MEAS        XXXX arcmin XXXX XXXX XXX  XXXX XXX
                                                    minutes
GPS2_HDG_MEAS  XXXX arcmin XXXX XXXX XXX  XXXX XXX  minutes
GPS_PSR1_MEAS  XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
GPS_PSR2_MEAS  XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
GPS_PSR3_MEAS  XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
GPS_PSR4_MEAS  XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
GPS_PSR5_MEAS  XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
GPS_PSR6_MEAS  XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
GPS_PSR7_MEAS  XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
GPS_PSR8_MEAS  XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
GPS_PSR9_MEAS  XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
GPS_PSR10_MEAS XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
GPS_PSR11_MEAS XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
GPS_PSR12_MEAS XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
LORAN_TDW_MEAS         XXXX metres XXXX XXXX XXX  XXXX XXX
                                                    minutes
LORAN_TDX_MEAS         XXXX metres XXXX XXXX XXX  XXXX XXX
                                                    minutes
LORAN_TDY_MEAS         XXXX metres XXXX XXXX XXX  XXXX XXX
                                                    minutes
LORAN_TDZ_MEAS         XXXX metres XXXX XXXX XXX  XXXX XXX
                                                    minutes
OMEGA_PD1_MEAS         XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
OMEGA_PD2_MEAS         XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
OMEGA_PD3_MEAS         XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
OMEGA_PD4_MEAS         XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
OMEGA_PD5_MEAS         XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
OMEGA_PD6_MEAS         XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
OMEGA_PD7_MEAS         XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
X_INS_POS_MEAS  XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
Y_INS_POS_MEAS  XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
Z_INS_POS_MEAS  XXXX metres XXXX XXXX XXX  XXXX XXX  minutes
X_INS_VEL_MEAS  XXXX m_per_s XXXX XXXX XXX  XXXX XXX  minutes
Y_INS_VEL_MEAS  XXXX m_per_s XXXX XXXX XXX  XXXX XXX  minutes
Z_INS_VEL_MEAS  XXXX m_per_s XXXX XXXX XXX  XXXX XXX  minutes
INS_ROLL_MEAS   XXXX arcmin XXXX XXXX XXX  XXXX XXX  minutes
INS_PITCH_MEAS  XXXX arcmin XXXX XXXX XXX  XXXX XXX  minutes
INS_HDG_MEAS    XXXX arcmin XXXX XXXX XXX  XXXX XXX  minutes

------------------------------------ LeverArmB_Vx(AllSensor)
--      Relative to SPERRY_INSs
INIT_LeverArmB_Vx
```

SPERRY_MK29_INS1	0.0	0.0	0.0	metres
LITTON_LTN90_INS1	0.0	0.0	0.0	metres
SIM_INS1	0.0	0.0	0.0	metres
SPERRY_MK29_INS2	0.0	0.0	0.0	metres
LITTON_LTN90_INS2	0.0	0.0	0.0	metres
SIM_INS2	0.0	0.0	0.0	metres
ASHTECH_XII_GPS	0.0	0.0	0.0	metres
TRIMBLE_TANS_PY_GPS1	-3.05	2.44	-4.88	metres
TRIMBLE_TANS_PY_GPS2	-3.05	2.44	-4.88	metres
TRIMBLE_VECTOR_GPS	0.0	0.0	0.0	metres
UofC_dGPS1	0.0	0.0	0.0	metres
UofC_dGPS2	0.0	0.0	0.0	metres
SIM_GPS	0.0	0.0	0.0	metres
INTERNAV_LC360_LORAN	-4.57	2.44	-4.88	metres
SIM_LORAN	0.0	0.0	0.0	metres
SIM_SPEED_LOG	0.0	0.0	0.0	metres
SIM_OMEGA	0.0	0.0	0.0	metres
UNKNOWN	0.0	0.0	0.0	metres

----------[Content-type: text/plain; name=Simulation.README]----------

Simulation configuration file (s*.cfg)

File names are (see SimSetup.cfg)

S12.cfg - INS1 & INS2
S1V1P2L.cfg - INS1, GPS1, GPS2, LORAN
S2V1P2L.cfg - INS2, GPS1, GPS2, LORAN
S1V1P2.cfg - INS1, GPS1, GPS2
S2V1P2.cfg - INS2, GPS1, GPS2
S1V1L.cfg - INS1, GPS1, LORAN
S2V1L.cfg - INS2, GPS1, LORAN
S1P2L.cfg - INS1, GPS2, LORAN
S2P2L.cfg - INS2, GPS2, LORAN
S1V1.cfg - INS1, GPS1
S2V1.cfg - INS2, GPS1
S1P2.cfg - INS1, GPS2
S2P2.cfg - INS2, GPS2
S1L.cfg - INS1, LORAN
S2L.cfg - INS2, LORAN

This example uses S1V1P2L.cfg

Line 1-6 SENSORS

Sensor description is divided into:
- . generic sensor, AllGenericSensors (SimConfig.spc)
- . sensor model, AllSensorModels (SimConfig.spc)
- . LINEAR/EXTENDED measurement model
- . UNCORRELATED/DECORRELATED
- . No measurement trigger time and unit

Line 8-16 SYSTEM_STATES

System States Enabled, AllSystemStates (SimTypes.spc)

Line 18-35 MARKOV_STATES

Markov States Enabled, AllMarkovStates (SimTypes.spc)

Line 37-48 MEASUREMENTS

Measurements Enabled, AllMeas (SimTypes.spc)

----------[Content-type: text/plain; name=s1v1p2l.cfg.v2.1]----------

```
SENSORS
--GenericSnsr   SensorModel    MeasModel MeasCorrelation NoMeasTimeTrigger
      INS1_SNSR  SPERRY_MK29_INS1        LINEAR     UNCORRELATED  1.0
                                                                  MINUTES
      GPS1_SNSR  TRIMBLE_TANS_PY_GPS1 LINEAR     UNCORRELATED  2.0
                                                                  MINUTES
      GPS2_SNSR  TRIMBLE_TANS_PY_GPS2 LINEAR     UNCORRELATED  2.0
                                                                  MINUTES
      LORAN1_SNSR INTERNAV_LC360_LORAN EXTENDED DECORRELATED
      5.0 MINUTES

SYSTEM_STATES

      X_POS_ERR
      Y_POS_ERR
      X_VEL_ERR
      Y_VEL_ERR
      X_ATT_ERR
      Y_ATT_ERR
      Z_ATT_ERR

MARKOV_STATES
```

X_ACCEL_BIAS_ERR
Y_ACCEL_BIAS_ERR
X_GYRO_BIAS_ERR
Y_GYRO_BIAS_ERR
Z_GYRO_BIAS_ERR
X_LOG_VEL_ERR
Y_LOG_VEL_ERR
X_GPS1_POS_ERR
Y_GPS1_POS_ERR
GPS1_TIME_LAG
X_GPS2_POS_ERR
Y_GPS2_POS_ERR
LORAN_TDW_ERR
LORAN_TDX_ERR
LORAN_TDY_ERR
LORAN_TDZ_ERR

MEASUREMENTS

X_LOG_VEL_MEAS
Y_LOG_VEL_MEAS
X_GPS1_POS_MEAS
Y_GPS1_POS_MEAS
X_GPS2_POS_MEAS
Y_GPS2_POS_MEAS
LORAN_TDW_MEAS
LORAN_TDX_MEAS
LORAN_TDY_MEAS
LORAN_TDZ_MEAS

APPENDIX C DESCRIPTION OF ENHANCED DIINS GENERIC ARCHITECTURE

The goal of this appendix is to define generic or general interfaces to the DIINS software to simplify the task of adapting the software to new interface requirements. A preliminary design of a generic architecture for DIINS and a detailed design of the input/output interfaces will be proposed in the sub-sections which follow. These design concepts will form the basis for the re-engineering of the DIINS XDM software to form the ADM software.

C.1 Preliminary Design

Figure C-1 illustrates the high level system/software design for a proposed enhanced DIINS architecture as would be used in an operational environment. The following sections describe the main components of the software and highlight changes from the current implementation that make the architecture more generic and easily adaptable to new input/output requirements.

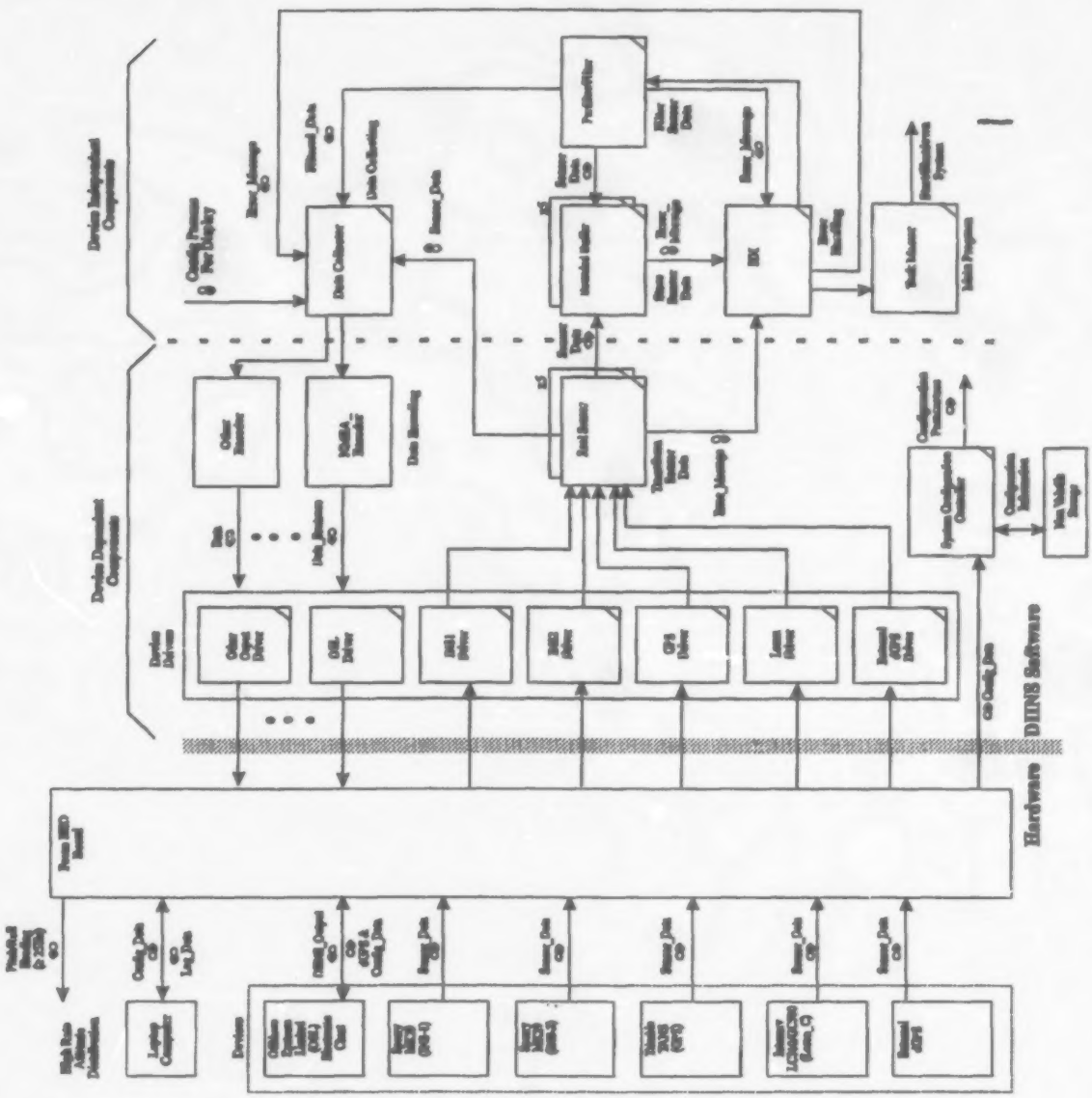

Figure C- 1 Enhanced DIINS Preliminary Design

C.1.1 Device Drivers

The OSL and sensor hardware interface to the DIINS software is via a ForceISIO VME RS232/RS422 interface board. A Device Driver component will be associated with each hardware device. Port and interrupt handling would be encapsulated within the Device Driver. Adding hardware devices to DIINS or changing interface boards will require the development of additional drivers or modification of existing drivers. The OSL driver outputs DIINS data to the OSL Electronic Chart for display. The sensor drivers read data from the sensors, via the ForceISIO board, and send the data on to the Real Sensor component associated with the driver.

Since the original DIINS Software Enhancement Study was conducted, some changes have been made to the code by the DREO staff in order to improve the portability of the code. One particular area where such an enhancement has been made is to change from using the Telesoft mail system to a DREO developed mail system for the internal transfer of data within the DIINS software. This mail system dependency was one of the original areas identified for improvement, and the original PRIOR study report proposed to replace the mail system by the use of the existing DIINS queue object. In light of the changes made by DREO the exact implementation of the inter-process communication in the DIINS ADM is still TBD. It may be appropriate to maintain the current new DREO implementation or there may be other factors that make the change to a conventional queue more appropriate/attractive. It is not necessary, and hence will not be attempted, to decide on this detailed implementation issue in Phase 1 of the ADM development. The wording extracted from the original study report which is presented in the following sections maintains the assumption that the mail system will be replaced by the queue. If the mail system is maintained, it can still be viewed that the mail system will be used to form the implementation of the queue. The architecture described in this appendix which is based on the use of a queue will be fundamentally unchanged whether the queue is implemented by a mail system or not.

The Mailbox mechanism which is used to pass data between the Device Driver and Real Sensor in the current implementation may be replaced by a queue mechanism encapsulated in the Real Sensor component, and this is the solution that is described in the following sections. This queue mechanism will be described in more detail in **Section C.2.**

C.1.2 Real Sensor

In the current implementation of DIINS, the Real Sensor software components are similar in structure and perform similar functions. The Real Sensor components in the enhanced design will be instantiations of a generic package. A Real Sensor package will be instantiated for each sensor connected to the DIINS software. The generic package will be described in more detail in **Section C.2.1.** The Real Sensor component will receive the raw sensor data from its associated device driver. The raw sensor data will be transformed into a record structure containing the sensor readings (e.g., Latitude, Longitude, Heading, etc.). The sensor readings will then be passed to the Data Collector and Bounded Buffer components of the system.

C.1.3 Bounded Buffer

The Bounded Buffer components of the system store the sensor readings received from the Real Sensors. As in the current implementation of DIINS, the Bounded Buffers will be instantiations of the generic queue package. A Bounded Buffer package will be instantiated for each sensor connected to the DIINS software.

C.1.4 Prefilter/Filter

The Prefilter/Filter reads the sensor data from the Bounded Buffer associated with each sensor. Filtered data is then passed to the Data Collector component. The Additional Secondary Factors routines in the current implementation of DIINS would be encapsulated in the Prefilter/Filter component in the enhanced design of DIINS. In the current developmental implementation of DIINS, the number of filters operated and their configurations, in terms of the sensor data included in the filters, is defined in a configuration file (refer to section 4). In the enhanced operational design of DIINS, this default configuration information will be stored in non-volatile storage, with the parameters being able to be displayed and modified via the user interface. The actual access to the configuration data within the system will be via a collection of functions which the System Configuration Controller can use to set or retrieve the configuration information. The use of functions allows the configuration parameters to be hidden within the body of a package, thus avoiding recompilation of the whole system when a default parameter is changed. The enhanced Prefilter/Filter design will be implemented so that only the configuration information need be modified if an additional sensor is connected to the DIINS software.

C.1.5 FDI

In the current implementation of DIINS, the FDI performs both hard and soft sensor failure detection and isolation. In addition to this capability, the enhanced FDI will handle software exceptions. When an exception is raised in one of the software components in the system, an error message will be generated identifying the exception and the source of the exception. The error will also be flagged as critical or non-critical. As an example, a Tasking Error exception raised in one of the system tasks would be considered a critical error but a Constraint Error exception on a sensor value would not. The error message will then be passed to the FDI. If the error is critical, the FDI will call the Task Master to initiate a shutdown of the system. An error message indicating the nature of the error, its location, and its criticality will be passed to the Data Collector component for display on the output interface. This ability to display a software error and the source of the error will support the maintenance of the DIINS software during operation.

C.1.6 Data Collector

The Data Collector component of the system receives sensor data from the Real Sensors, filtered data from the Prefilter/Filter, and error messages from the FDI. This information is then passed to the NMEA Encoder for encoding. A more detailed design description of the Data Collector component is contained in **Section C.2.2**.

C.1.7 NMEA Encoder

The NMEA Encoder component of the system would receive sensor data and error messages from the Data Collector component. This information is then formatted into data sentences to be passed through the OSL device driver to the OSL Electronic Chart for display. Changes to the output format or additional data sentences will require modifications to existing routines or additional formatting routines within the NMEA Encoder component. A more detailed design description of the enhanced NMEA Encoder component is described in **Section C.2.2**.

C.1.8 Task Master

The Task Master component controls the overall operation of the DIINS software. On start up, Task Master will instantiate the appropriate number of generic Real Sensor and Bounded Buffer packages (based on information from the system configuration package) and initiate system start up. Task Master will also perform an orderly shutdown of the system when a critical error is detected (Note: The critical error message would be output before DIINS software is terminated.).

C.2 Detailed I/O Interface Design

C.2.1 Input Interface

C.2.1.1 Analysis of Current Design

Figure C-2 to **Figure C-4** illustrate the detailed design of the Real Sensor components in the current implementation of the DIINS software. Analysis of these components reveals a similarity in their structure and function. The DCT tasks (Sperry_MK29_1_DCT, Sperry_MK29_2_DCT, Internav_LC360_DCT, and Trimble_TANS_PY_DCT) and Sensor packages (Sperry_MK29_1, Sperry_MK29_2, Internav_LC360, and Trimble_TANS_PY) can be combined to form a generic input interface for the DIINS software.

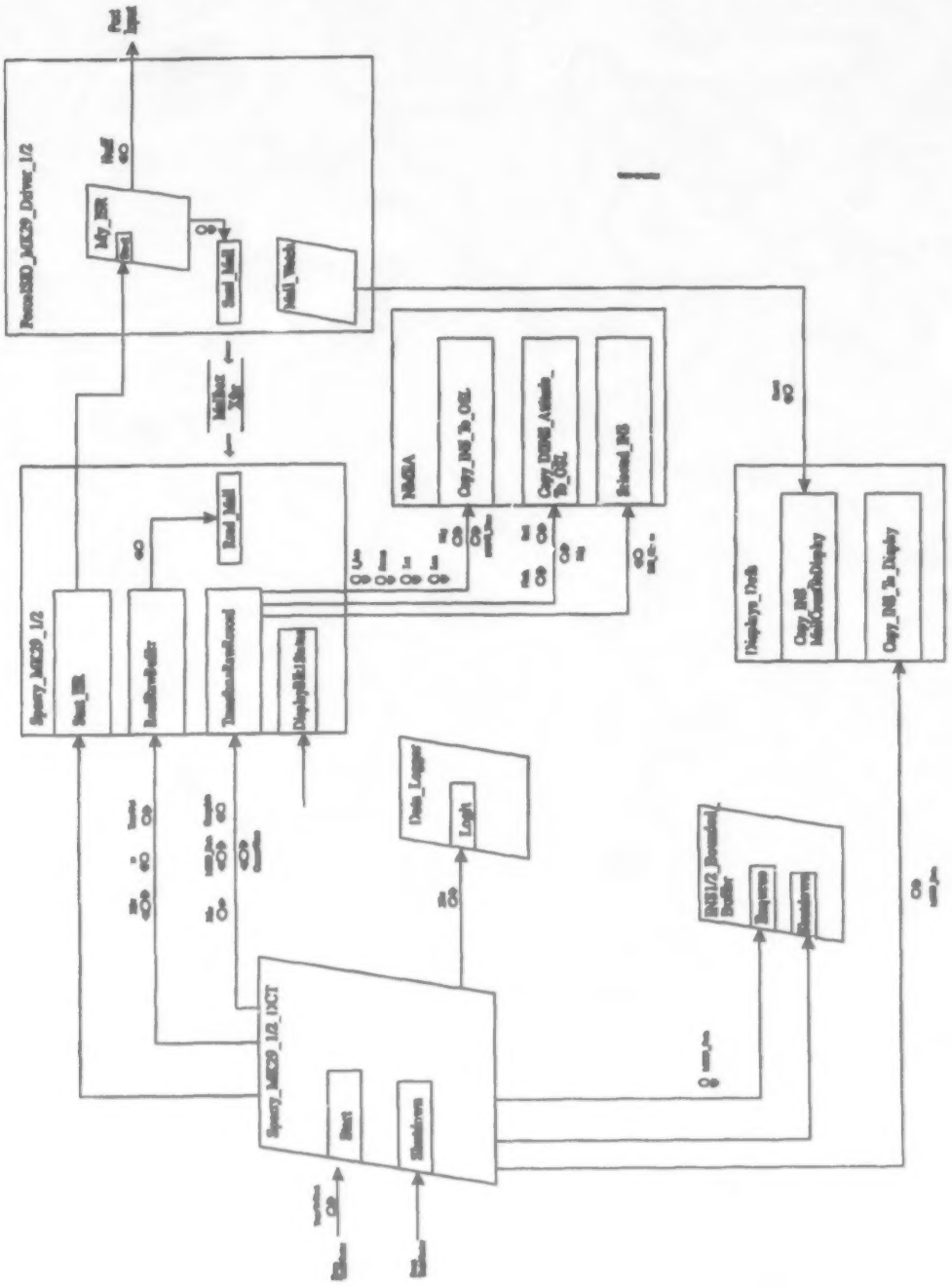

Figure C- 2 Sperry MK29 1 & 2 (INS)

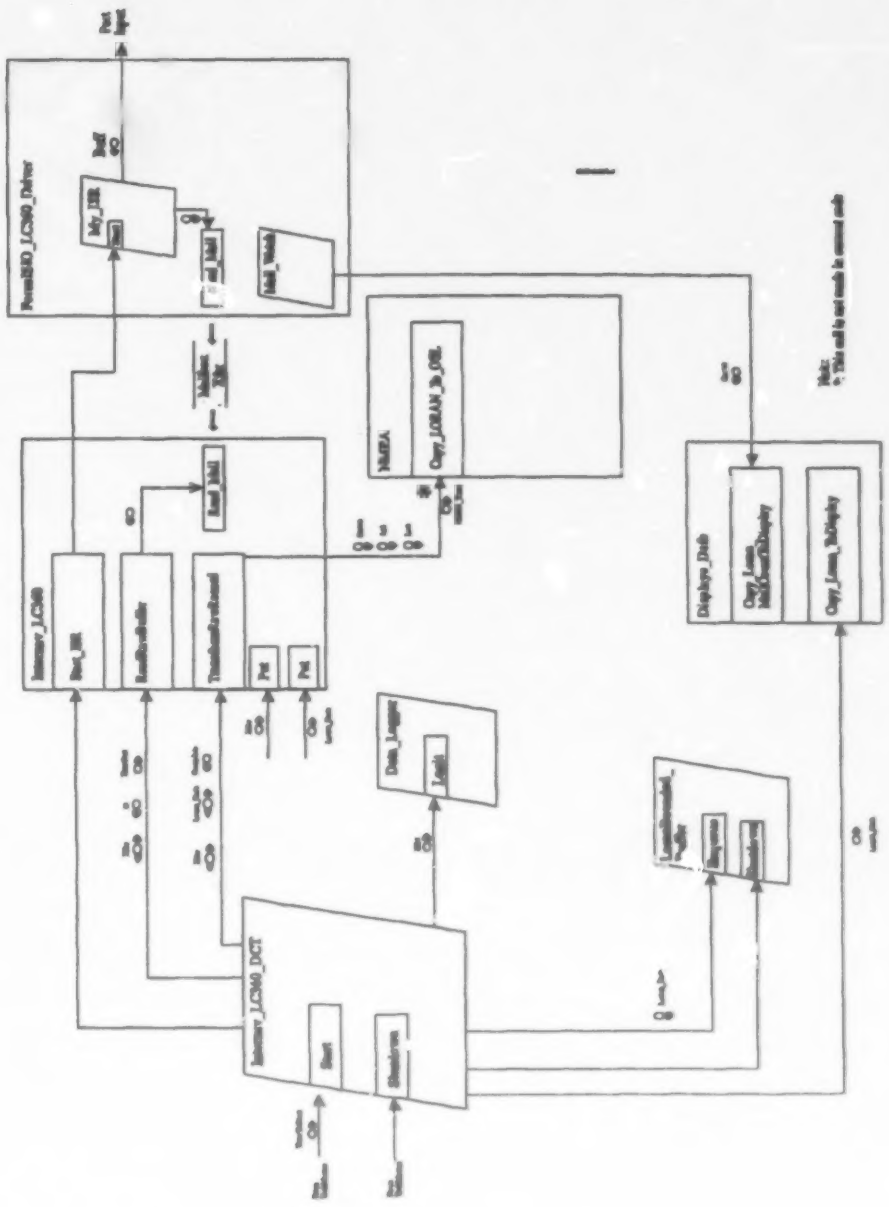

Figure C- 3 Internav LC360 (Loosely)

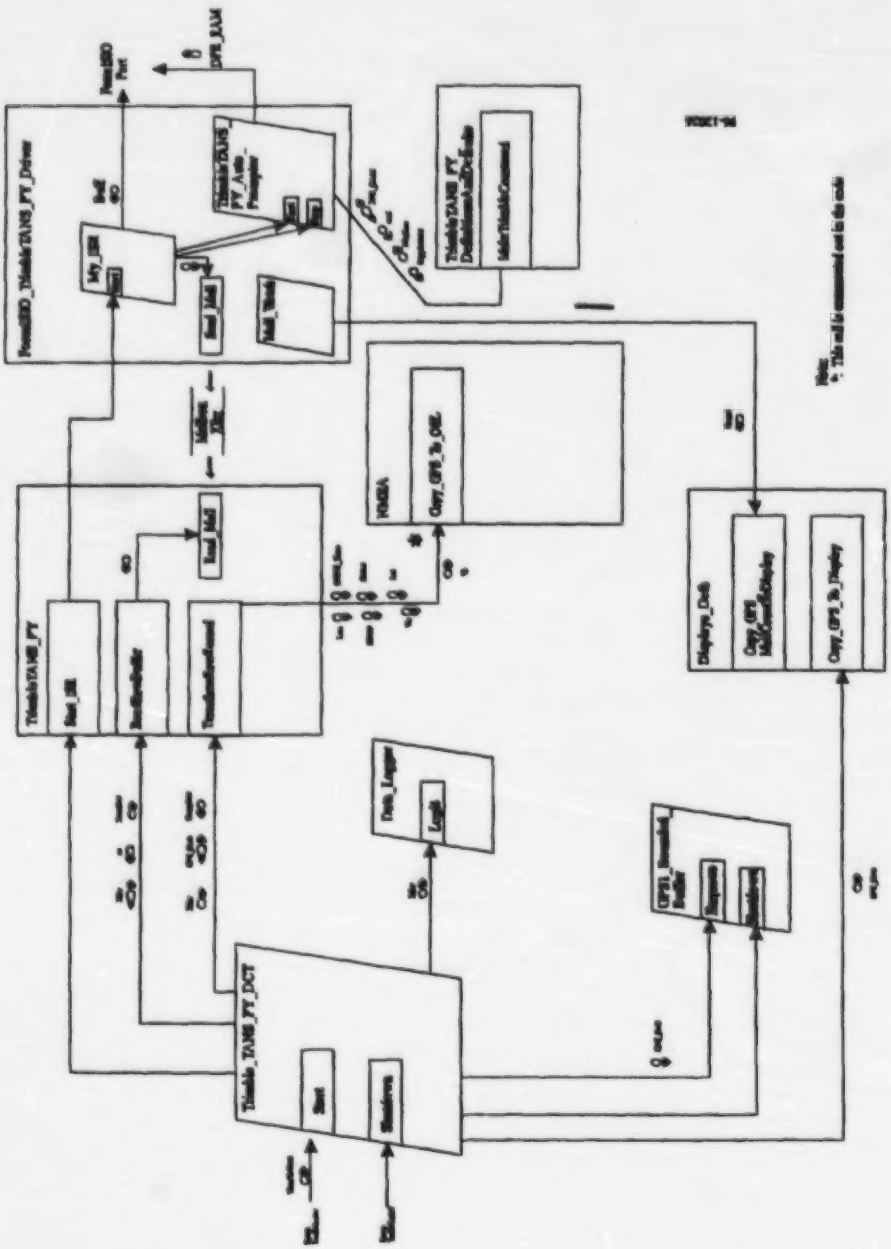

Figure C- 4 Trimble TANS PY (GPS)

C.2.1.2 Enhanced Design

Figure C-5 illustrates the detailed design for a generic Real Sensor package. In this design the Read_Raw_Buffer procedure receives raw sensor data from the associated device driver. This data is placed in a queue. The Queue is an instantiation of the generic queue package. The Sensor_Master task will wait for data on the queue. When data is available, the Sensor_Master will retrieve the raw sensor data from the queue. The Transform_Raw_Record procedure inside the Sensor_Master task will perform calculations on the raw sensor data and transform the data into a sensor data record. The sensor data record will then be passed to the appropriate Bounded Buffer package and to the Data Collector package. Any error messages generated from the Real Sensor package will be passed to the FDI package. The package's tasks will be hidden within the body of the package. Encapsulation of tasks reduces the amount of extraneous information in the package specification and conceals task entries that should not be made public. The Start and Shutdown procedures will control access to the Start and Stop entries in the tasks. The Start procedure will rendezvous with the Start entries in the tasks. If a task fails to start (i.e., an exception is raised), a false status will be returned to the Task Master. The Shutdown procedure will rendezvous with the Stop entries in the tasks.

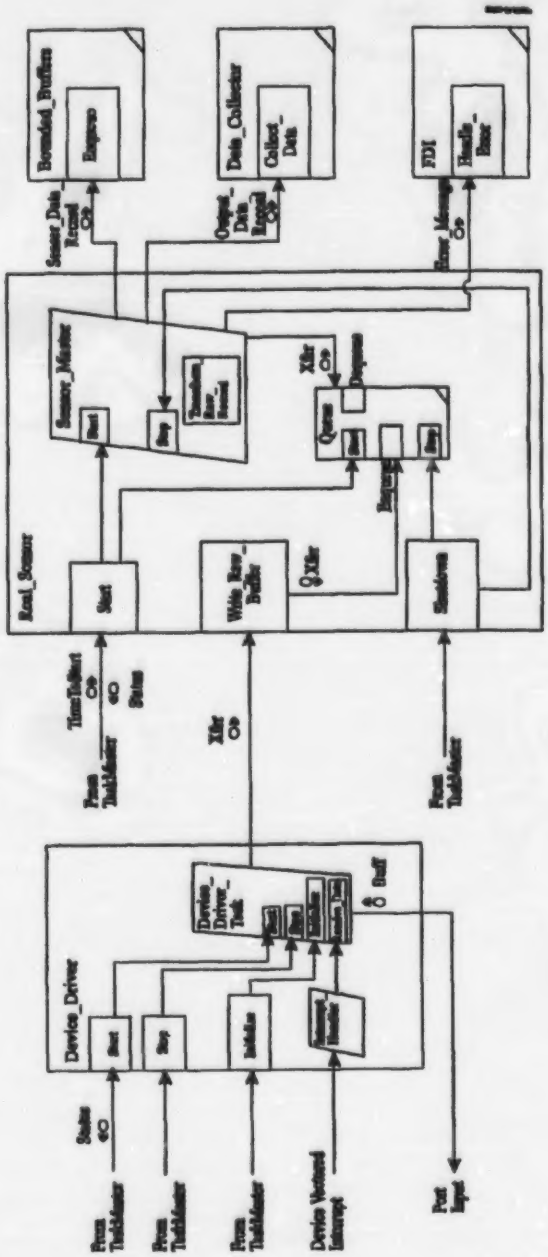

Figure C- 5 Real Sensor Generic

The specification for this package would resemble the following generic specification:

```
generic

   -- Data type for raw sensor data from device driver.

   type Raw_Sensor_Data_Type is private;

   -- String identifier for sensor.

   Sensor_Identifier : in String;

   -- Procedure to transform raw sensor data.

   with procedure Transform_Raw_Record
      (Raw_Sensor_Data : in Raw_Sensor_Data_Type);

package Real_Sensor is

   ------------------------------------------------
   -- Initialize sensor package.
   ------------------------------------------------

   procedure Start (TimeToStart : in     Duration;
                    Status      :    out Boolean);

   ------------------------------------------------
   -- Read raw sensor data from device driver.
   ------------------------------------------------

   procedure Read_Raw_Buffer
      (Raw_Sensor_Data : in Raw_Sensor_Data_Type);

   ------------------------------------------------
   -- Shutdown sensor package.
   ------------------------------------------------

   procedure Shutdown;

end Real_Sensor;
```

To instantiate this package, a data type for the raw sensor data and a procedure to transform the raw sensor data must first be defined. The data type and procedure would then become input parameters in the instantiation of the generic. A string identifier for the sensor is also passed in

as a parameter to be used in the generation of any error messages. As an example, an instantiation of the Real Sensor package for an INS sensor would resemble the following:

```
type XferREC is
   record
      Time    : Duration;
      BuffLen : Integer;
      Buffer  : BuffArray;
   end record;

procedure Transform_Raw_INS is

   begin
      .
      .
      .
   end Transform_Raw_INS;

package Real_INS is new Real_Sensor
   (XferREC, "INS1", Transform_Raw_INS);
```

C.2.2 Output Interface

C.2.2.1 Analysis of Current Design

Figure C-6 illustrates the detailed design of the NMEA or output interface component in the current implementation of the DIINS software. Analysis of this component reveals two basic requirements on this portion of the DIINS software. The first requirement is for this component to receive sensor data. The second requirement is for this component to encode or format the sensor data it receives into data sentences to be output over the OSL interface.

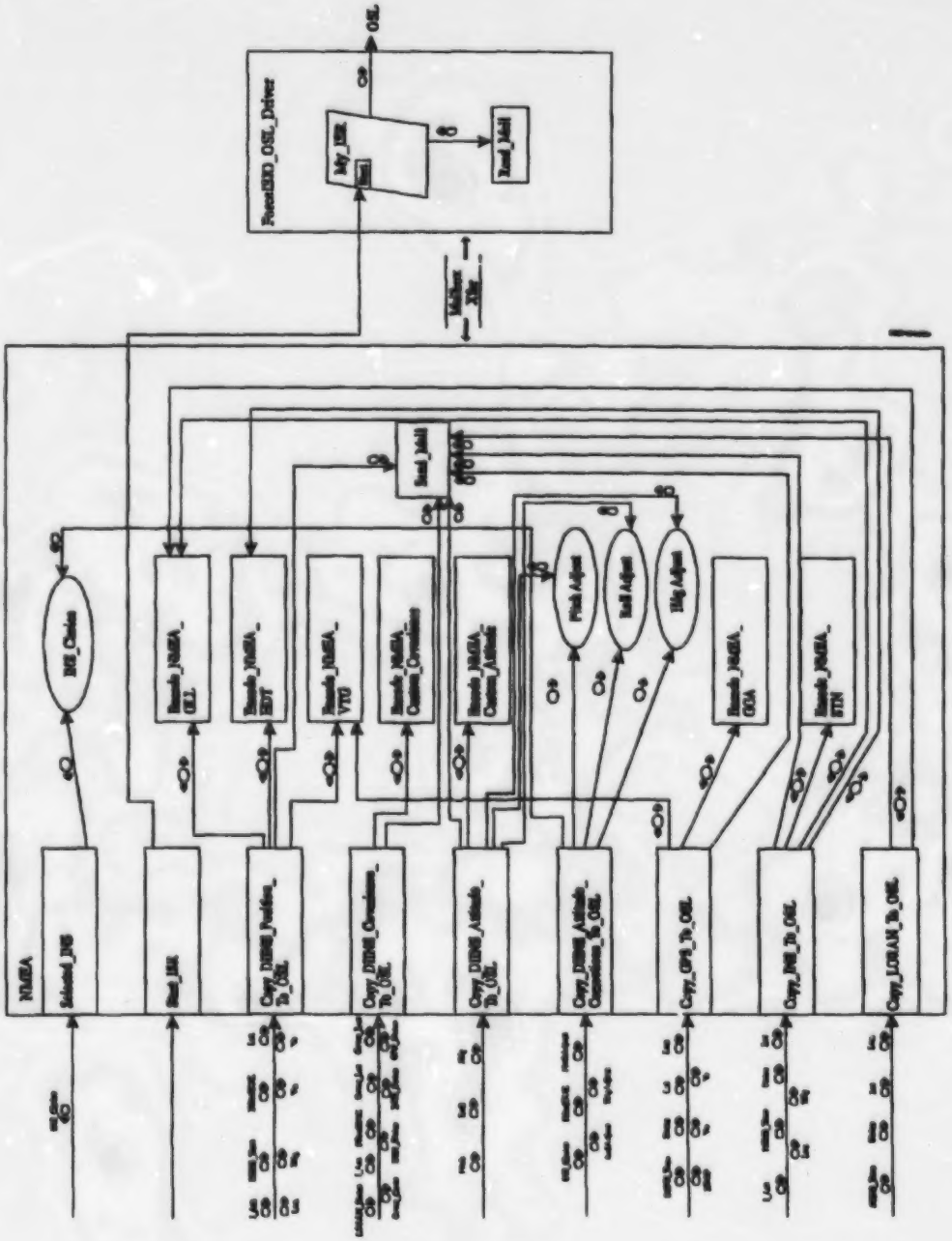

Figure C- 6 NMEA Output Interface

C.2.2.2 Enhanced Design

Figure C-7 illustrates the detailed design for an enhanced output interface. This design is a "cleaned-up" or simplified version of the current design. The two basic functions of the output interface have been captured in the Data Collector and NMEA Encoder packages. By isolating these functions in the design the resulting code will be easier to adapt to meet new or changing requirements. The Collect_Data procedure in the Data Collector package receives sensor data or error messages in the form of a variant record containing the individual sensor data elements or error message string. The data record is placed in a queue. The Queue is an instantiation of the generic queue package. The Collector_Master task will wait for data on the queue. When data is available, the Collector_Master will retrieve the data record from the queue and pass it to the Encode procedure in the NMEA Encoder package (and to other encoder packages as required) to be formatted into a data sentence. The routines to format the individual sensor data elements will be encapsulated within the procedure Encode. The main portion of the Encode procedure will use these routines to construct the data sentences to be sent to the OSL Driver. The Mailbox mechanism which is used to pass data between the NMEA and OSL Driver in the current implementation may be replaced by a queue mechanism encapsulated in the OSL Driver, as indicated in this appendix.

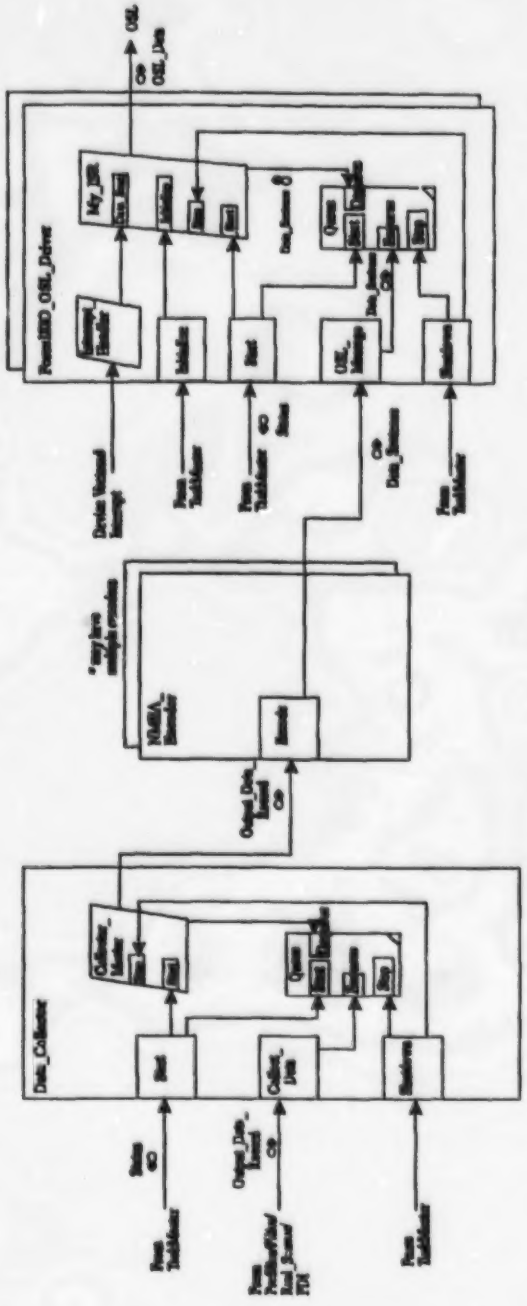

Figure C- 7 Enhanced Output Interface

The specifications for the Data Collector and NMEA Encoder packages would resemble the following:

```
with Global_Definitions;

package Data_Collection_Records is

   type Output_Data_Type is
      (Void_Data,
       Sensor_Data,
       Filter_Data,
       Error_Data);

   type Output_Record_Type
      (Output_Data : Output_Data_Type := Void_Data) is
      record
         case Output_Data is

            when Void_Data =>
               null;

            when Sensor_Data =>
               Sensor :
                  Global_Definitions.Sensor_Data_Type;

            when Filter_Data =>
               Filter :
                  Global_Definitions.Filter_Data_Type;

            when Error_Data =>
               Error_Message :
                  Global_Definitions.Error_Message_Type;
         end case;

      end record;

end Data_Collection_Records;


with Data_Collection_Records;

package Data_Collector is

   ------------------------------------------------
   -- Initialize data collector package.
```

```
   ------------------------------------------------
   procedure Start (Status : out Boolean);
   ------------------------------------------------
   -- Shutdown data collector package.
   ------------------------------------------------

   procedure Shutdown;

   ------------------------------------------------
   -- Receive output data.
   ------------------------------------------------

   procedure Collect_Data
      (Output_Data_Record : in Data_Collection_Records.
          Output_Record_Type);

end Data_Collector;


with Data_Collection_Records;

package NMEA_Encoder is

   ------------------------------------------------
   -- Encode output data.
   ------------------------------------------------

   procedure Encode
      (Output_Data_Record : in Data_Collection_Records.
          Output_Record_Type);

end NMEA_Encoder;
```

C.3 Conclusions

This section lists the main conclusions drawn from the analysis of the ADM software architecture during the Phase 1 Systems Analysis:

a. Implementing the proposed design for the enhanced DIINS architecture will enhance the maintainability and flexibility of the software, thus making it easier to add new sensors to DIINS or change DIINS output requirements.

b. Using the enhanced DIINS design, a sensor can be added by making the following modifications to the DIINS software:

- Create a device driver to drive the sensor input.
- Instantiate a Real Sensor package for the new sensor.
- Instantiate an additional Bounded Buffer package to store the data from the new sensor.
- Modify the system configuration information so that the Task Master and Prefilter/Filter will handle the additional sensor packages and sensor data.
- If necessary, add a new sensor data record type.
- If necessary, modify or create new encoding routines to format the new sensor data for output.

c. Using the enhanced DIINS design, a change to the DIINS output can be handled by making the following modifications to the DIINS software:

- Modify or create the encoding routines to handle the new output format.
- If necessary, modify the output interface driver.

d. The ability to display a software error and the source of the error will support the maintenance of the DIINS software during operation.

APPENDIX D HCI PROTOTYPE FORMS

The following sections briefly describe the HCI interface forms presented in this appendix.

The forms presented are primarily for use by the maintenance engineer, with only the form designed for selecting the sensors to be used for the navigation solution being specifically designed for use by the DIINS navigation operator.

Notes:

1. With all of the HCI forms shown in this appendix the data for a particular form is not entered until the "OK" command is entered. If it is required to cancel the data entry for a particular form the "Cancel" command can be used on the form.

2. The general features of windowing interface HCIs will be available on the DIINS HCI. These will include:

   - Selection and deselection of list items via mouse clicks;
   - Alphabetical finding of list entries by clicking on the list and entering the first letter of the item to be found;
   - Scrolling up and down list items, and their associated data items using the keyboard arrow keys; and
   - Tabbing between the various fields presented on the forms using the keyboard "Tab" key.

3. The GUI forms presented in this appendix were developed using Microsoft Visual Basic for the Windows 95 environment. The OSL SHINNADS PC supports a version of OS/2 which currently only allows Windows 3.1 (16 bit) applications to be run. Hence, depending on the development environment used to develop the actual forms for the DIINS ADM and the type and version of the operating system fitted in the SHINNADS PC, the look of some of the forms may vary from those presented in this appendix.

Figure D-1 Top Level DIINS HCI Form

This form allows the system user to either select the form to be used to select the sensors for the navigation solution, or to select to display the menu of options available for maintaining DIINS.

Figure D-2 Sensor Selection Form

This form allows the operator to select which sensors will be used to generate the navigation solution using DIINS. Only the sensors previously physically installed and installed in the software

using the Sensor Installation Form (Figure D-4) show up as being available for selection for the navigation solution.

Figure D-3 DIINS Maintenance Mode Menu Selections

This form is presented when "Maintenance" is selected on the top-level-form (Figure D-1). If "OperationalMode" is selected the form display reverts back to the top level form again (Figure D-1). The following options are available for selection on this form, and when selected cause the display of the associated lower level forms to enable the operator to review and modify various system parameters:

- Select Sensor
- Operational Mode
- Install Sensors
- Lever Arms
- Communications Setup
- FDI Parameters
- Logging Setup
- Failure Inject
- System Status

The prototype HCI also includes an alternative look-and-feel form for the entry of Lever Arm and Communications Setup information.

Figure D-4 Sensor Installation Form

This form presents the user with a list (on the left) of the available sensor models programmed into the DIINS software. The user (normally the maintenance engineer) selects and installs the sensors using this form to comply with the sensor configuration that is physically wired to the DIINS hardware. If the user makes a mistake or the sensor becomes unavailable the installed sensor can also be removed using this form. This form also allows the user to set or clear a check box which controls the distribution of the high rate attitude information from DIINS to the other ship systems.

Figure D-5 Sensor Lever Arm Setup Form

Before using DIINS to provide a navigation solution it is necessary to inform DIINS of the locations of the sensors relative to the reference point used within DIINS. Using this form the operator selects the sensor for which data is to be reviewed/modified and enters new data as required.

For the DIINS ADM the Reference point equals the Center of Floatation (CoF)

"Xc: distance FORWARD of reference point (negative for Aft)"
"Yc: distance ABOVE reference point(negative for below)"
"Zc: distance STARBOARD of reference point (negative for Port)"

The software would then have to translate these to DIINS coordinates using the Equations D-1, where Xd, Yd and Zd represent the DIINS coordinates:

$$Xd = Xc \qquad \text{(Eqns D-1)}$$
$$Yd = Zc$$
$$Zd = -Yc$$

Note:

A feature that could be considered for addition, provided it is in keeping with project budgets and time scales, would be the ability to change the reference point. For example once the data is entered, the default reference point could be changed to INS1, and the lever arms would be recalculated automatically. Note that the DIINS output is always referenced to the (0,0,0) of the coordinate frame, so if (0,0,0) is the CoF, then that is the reference point. If (0,0,0) is INS1, then the outputs are all referenced to INS1, and so on.

Figure D-6 Sensor Communication Channel Setup Form

DIINS communicates with SHINNADS and the other ship systems using serial communication interfaces, e.g. RS-232/422. This form allows the user to review and modify the parameters associated with the communications channel for the selected sensor. The available channel options are presented via drop down combo boxes, minimizing the data entry/checking requirements.

Figure D-7 FDI Parameters Setup Form

This form allows the system user to set up various initial values to be used by the FDI algorithms to detect hard and soft failures.

Figure D-8 Data Logging Setup Form

This forms allows the user to select the various data items to be logged during the operation of DIINS. A check box is provided for each sensor to select whether the sensor is to be logged or not. If the sensor is not to be logged, its Log Interval and Nominal Bytes/Second display fields are set to zero. A list of the available filters is provided to allow the user to select which filter outputs should be logged.

The form indicates the number of bytes per day that would be logged with the current form selections. This enables the user to determine how long the available storage volume for logging will last, or to be able to select the sensor and filter data to be logged to approximate a specific bytes/day requirement. There are two logging rate scale factors available on the form. The "Overall Rate Scale Factor" (bottom left) allows the user to adjust the overall logging rate by increasing the period at which the data is logged. Settings of the Overall Rate Scale Factor are 1.0 (Nominal value), 0.5, 0.25 and 0.1 (i.e. log at 1/10 th the rate of the nominal logging). In addition to the overall rate scale factor, each sensor has an individual "Rate Scale" factor that can be used to override the overall scale factor for the selected sensor. If the individual Rate Scales have been set and the Overall Rate Scale Factor is then modified, this resets all of the individual Rate Scale factors to the overall scale factor again. The form also displays (middle right) the number of bytes per second currently being logged for the filters with the current filter selections. The period at which the filter is updated is also displayed on the lower right of the form.

Figure D-9 Failure Injection Form

This form enables the DIINS sensor failure detection mechanisms to be verified to be operational. Both hard (complete failure of the sensor) and soft (small deviations in the sensor outputs due to sensor failure/degradation) failures can be injected, and the failures can be timed to occur either at a particular time (absolute timing) or relative to the current time. The duration of the injected failure can also be programmed via the form. The failure injection capabilities will be prohibited when the system is in operational mode, i.e. when the form of Figure D-1 is displayed or when the same form is displayed in icon style.

Figure D-10 System Status Form

This form enables the DIINS user to monitor the load currently being placed on the DIINS processor by the processing taking place in the current system configuration, e.g. which may include the operation of data logging. The form indicates which filter is currently selected by the DIINS FDI for provision of the navigation solution. The form also displays a dynamically updated list of software exception events, FDI Log and Filter Log text messages. Note, FDI and Filter logged data in general will not be displayed in the event list on this form. Only specific text messages that will be included in the logged data to support the monitoring of the correct operation of the FDI and Filter processing functions in general will be displayed. During nominal system operation no software exception events should be shown in the list. The exceptions raised by the software may be due to either software or hardware failures. The list of exceptions and log events can be cleared by the user.

Figure D-1 Top Level DIINS HCI Form

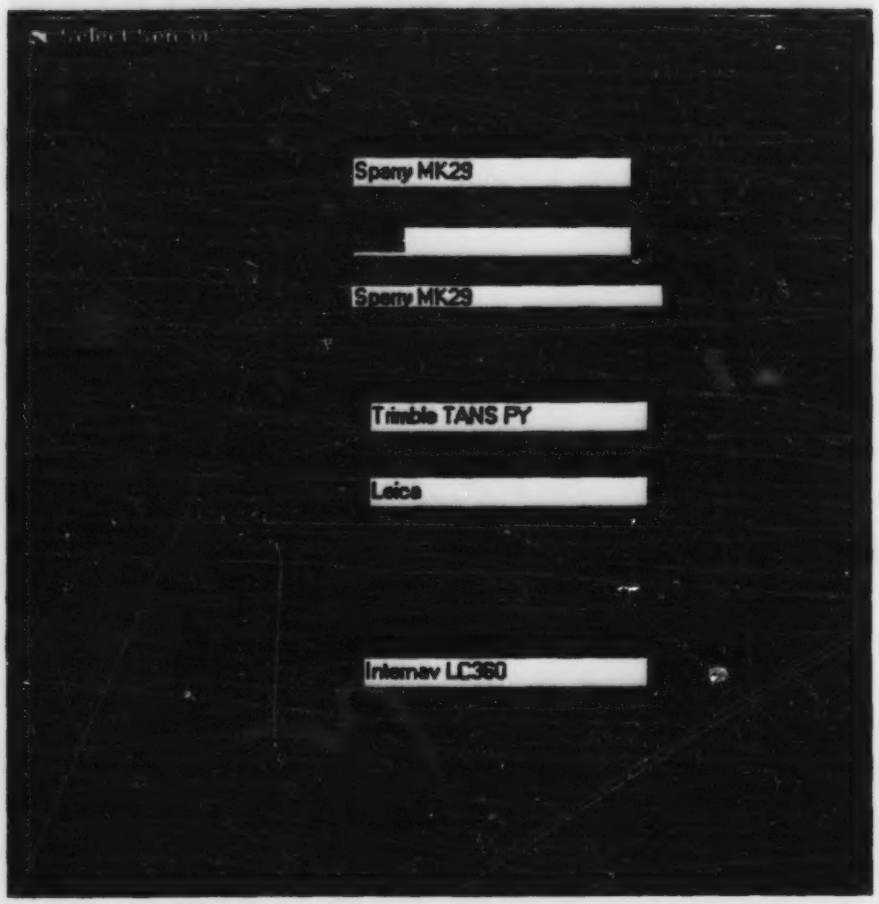

Figure D-2 Sensor Selection Form (from installed sensors)

Figure D-3 DIINS Maintenance Mode Menu Selections

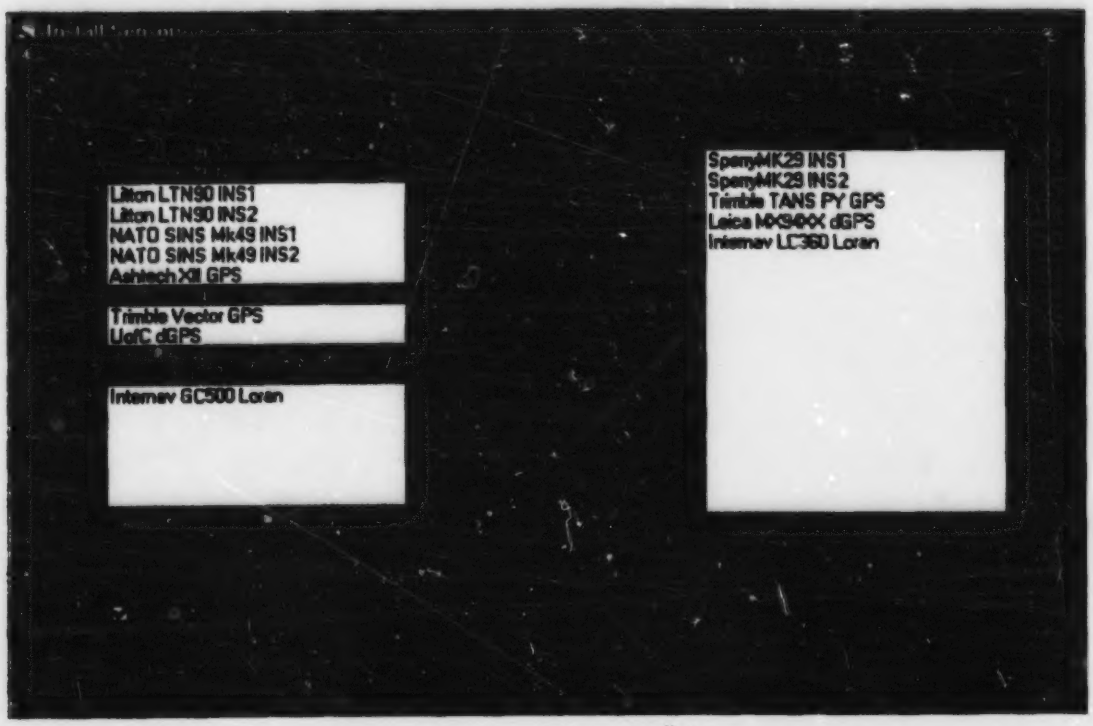

Figure D-4 Sensor Installation Form

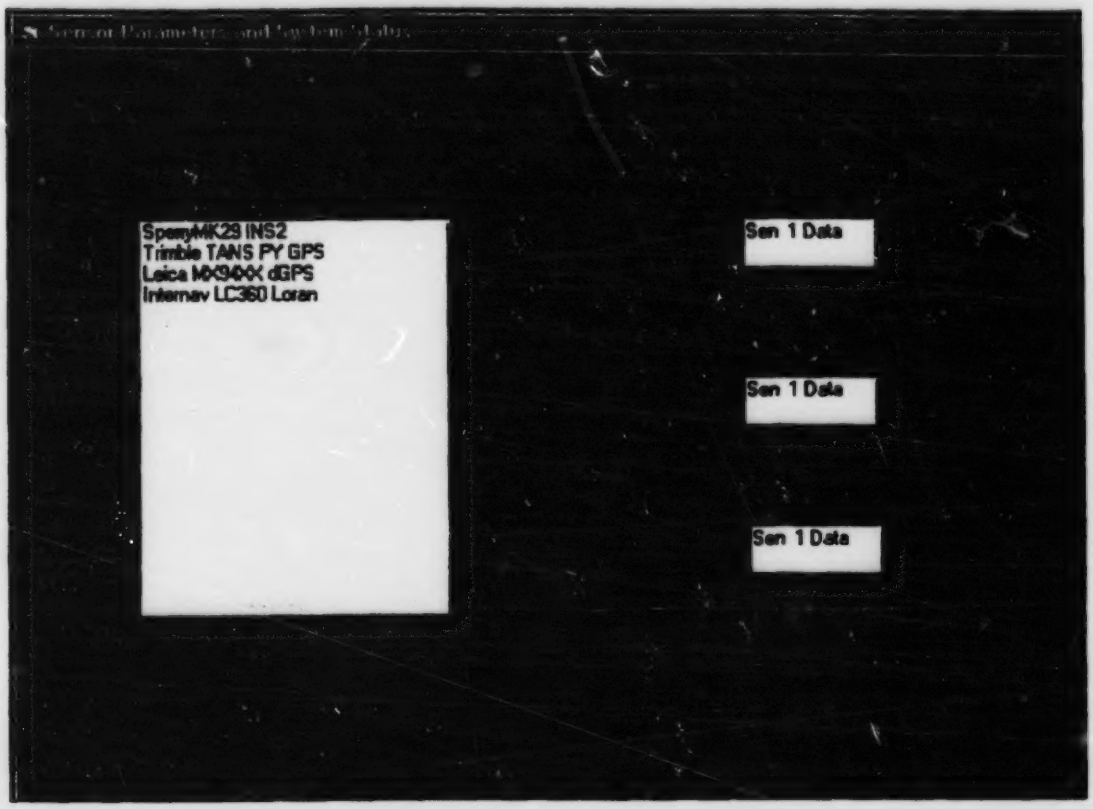

Figure D-5 Sensor Lever Arm Setup Form

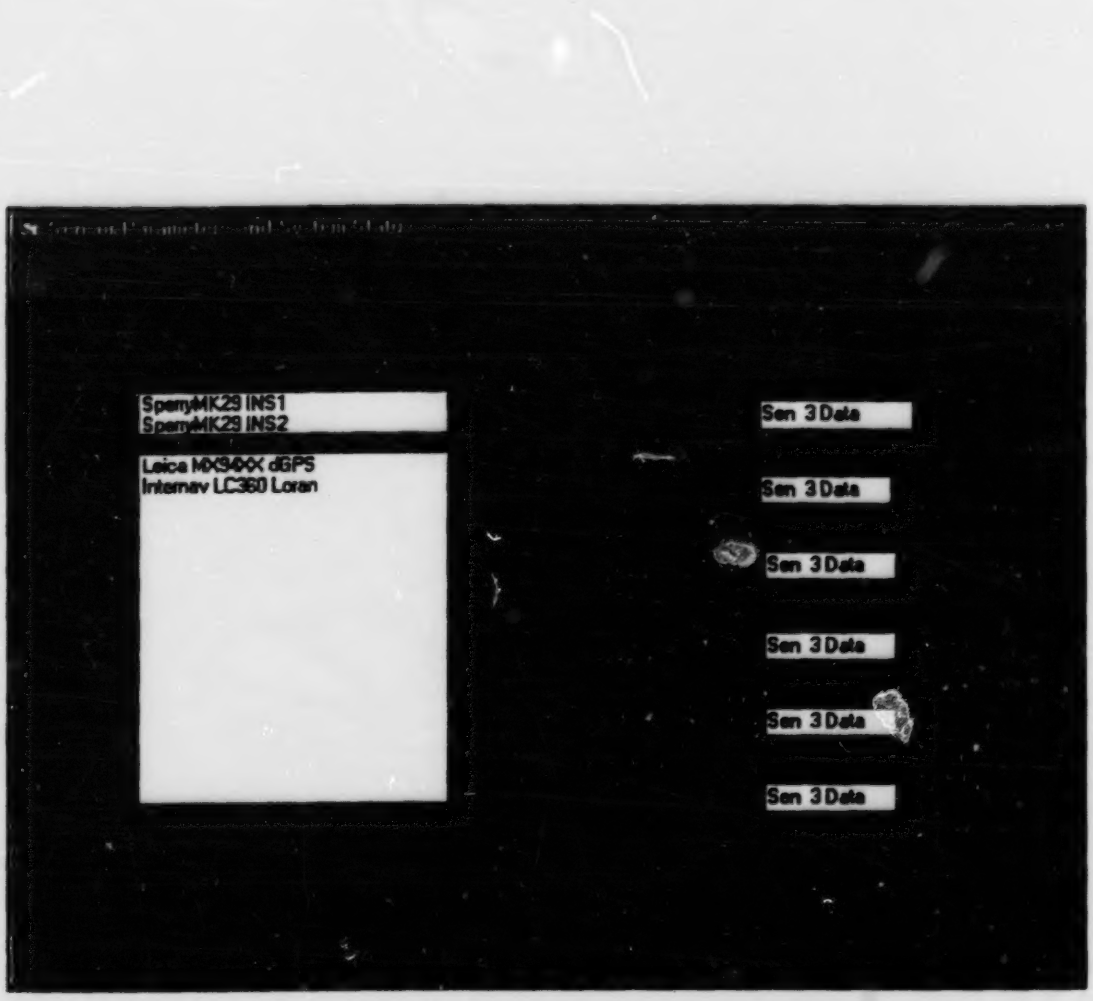

Figure D-6 Sensor Communication Channel Setup Form

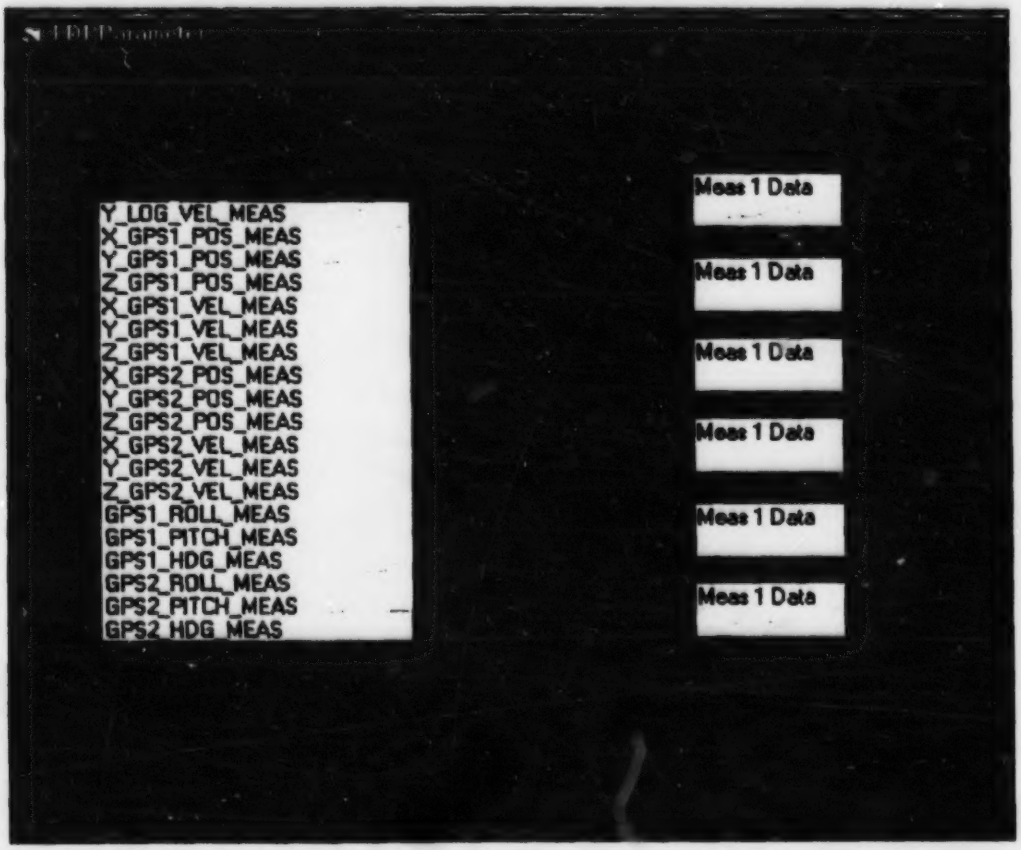

Figure D-7 FDI Parameter Setup Form

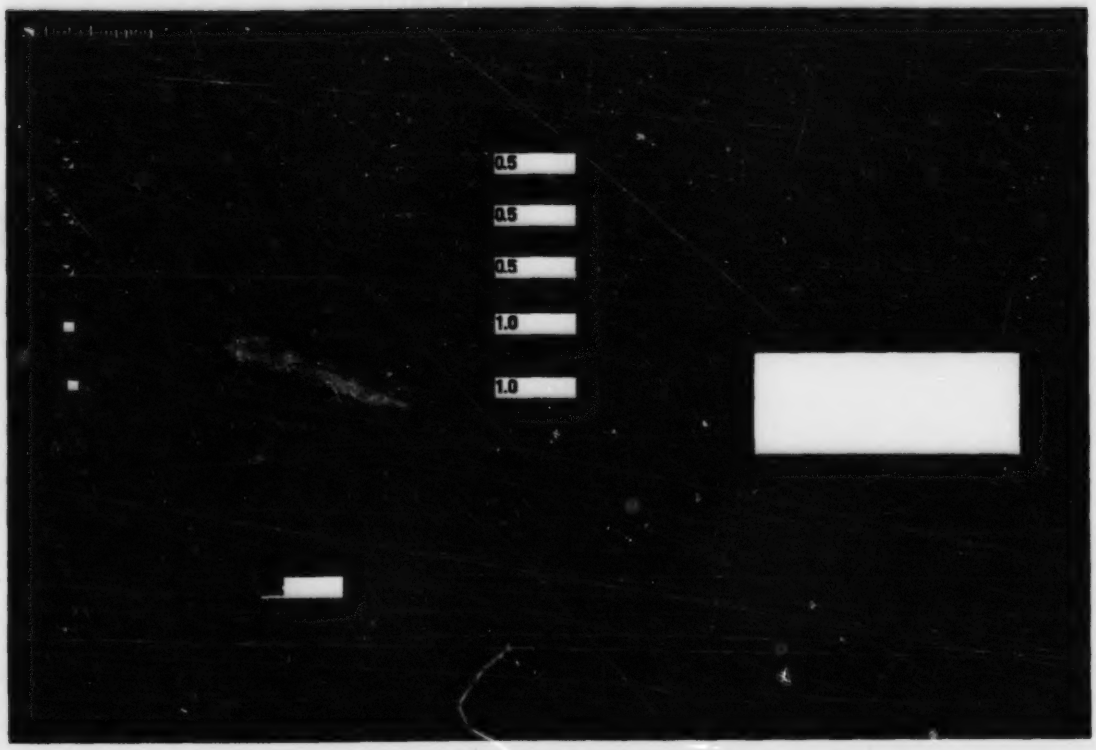

Figure D-8 Data Logging Setup Form

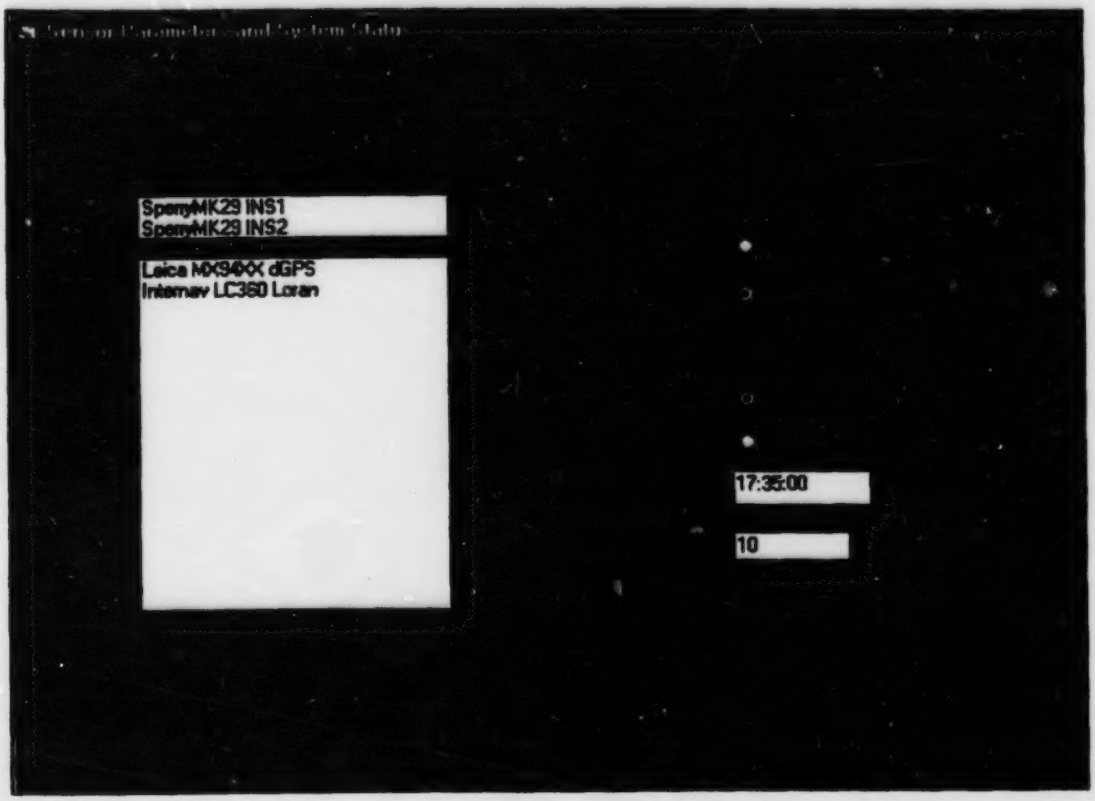

Figure D-9 Failure Injection Form

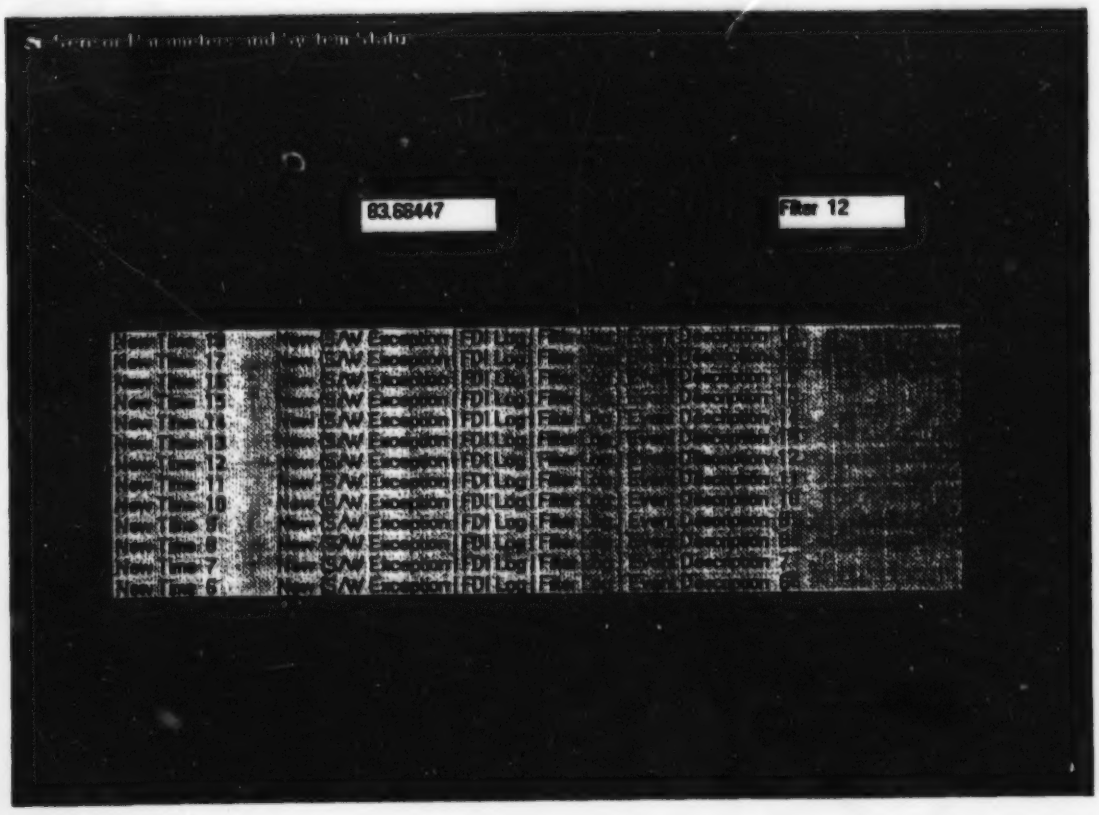

Figure D-10 System Status Form

APPENDIX E DATA LOGGING REQUIREMENTS

This appendix indicates (based on the data logging performed in the DIINS XDM) the nominal number of bytes per second required to be logged in the DIINS ADM. This data logging requirement represents a maximum to the amount of data that can be logged from the DIINS ADM. In practice the system user will be able to vary the volume of data logged using the data logging options available via the DIINS Human-Computer Interface Data Logging form.

The DIINS DataLogger logs both Raw Sensor and Filter data. The following sections indicate the format in which it is planned to log the various data items within the DIINS ADM.

Raw Data breakdown is as follows:

. GPS (NMEA) per 1.0 sec
"$R2,<DIINS time><Length of ZDA data><data><CR><LF>"
=> 4 + 8 + 2 + 37 + 2 = 53
"$R2,<DIINS time><Length of GGA data><data><CR><LF>"
=> 4 + 8 + 2 + 66 + 2 = 82
"$R2,<DIINS time><Length of VTG data><data><CR><LF>"
=> 4 + 8 + 2 + 38 + 2 = 54

==========> 189 bytes per 1.0 sec

. dGPS (NMEA) per 1.0 sec
"$R4,<DIINS time><Length of ZDA data><data><CR><LF>"
=> 4 + 8 + 2 + 37 + 2 = 53
"$R4,<DIINS time><Length of GGA data><data><CR><LF>"
=> 4 + 8 + 2 + 66 + 2 = 82
"$R4,<DIINS time><Length of VTG data><data><CR><LF>"
=> 4 + 8 + 2 + 38 + 2 = 54

==========> 189 bytes per 1.0 sec

. INS1 per 1.28 sec
16 X "$R0,<DIINS time><Length of data><data><CR><LF>"
=> 16 x (4 + 8 + 2 + 33 + 3)

==========> 800 bytes per 1.28 sec

. INS2 per 1.28 sec
16 X "$R1,<DIINS time><Length of data><data><CR><LF>"
=> 16 x (4 + 8 + 2 + 33 + 3)

==========> 800 bytes per 1.28 sec

. Loran per 1.0 sec
"$R3,<DIINS time><Length of data><data><CR><LF>"
=> 4 + 8 + 2 + 80 + 2

==========> 96 bytes per 1.0 sec

Sub-Total: Approx ~~41 MByte/Day~~ 142 Mb/day

Filter Data breakdown is a follows:

. Filter (1 to 15) per 1.28 sec (highest rate)

15 x "$P<FilterID>1,<Display data><CR><LF>"
=> 15 x (4 + 8 + 2 + 11x8 + 2)
=> 1560 bytes

"$P<FilterID>2,<State data><CR><LF>"
=> 4 + 8 + 2 + (1+NumStates+NumStates+1)x8 + 2

FilterID - NumStates => size

1 - 14	=> 4 + 8 + 2 + (1+14+14+1)x8 + 2 = 256
2 - 23	=> 4 + 8 + 2 + (1+23+23+1)x8 + 2 = 400
3 - 23	=> 4 + 8 + 2 + (1+23+23+1)x8 + 2 = 400
4 - 19	=> 4 + 8 + 2 + (1+19+19+1)x8 + 2 = 336
5 - 19	=> 4 + 8 + 2 + (1+19+19+1)x8 + 2 = 336
6 - 21	=> 4 + 8 + 2 + (1+21+21+1)x8 + 2 = 368
7 - 21	=> 4 + 8 + 2 + (1+21+21+1)x8 + 2 = 368
8 - 21	=> 4 + 8 + 2 + (1+21+21+1)x8 + 2 = 368
9 - 21	=> 4 + 8 + 2 + (1+21+21+1)x8 + 2 = 368
10 - 17	=> 4 + 8 + 2 + (1+17+17+1)x8 + 2 = 304
11 - 17	=> 4 + 8 + 2 + (1+17+17+1)x8 + 2 = 304
12 - 17	=> 4 + 8 + 2 + (1+17+17+1)x8 + 2 = 304
13 - 17	=> 4 + 8 + 2 + (1+17+17+1)x8 + 2 = 304
14 - 18	=> 4 + 8 + 2 + (1+18+18+1)x8 + 2 = 320
15 - 18	=> 4 + 8 + 2 + (1+18+18+1)x8 + 2 = 320

=> 5056 bytes

"$P<FilterID>3,<Residual & Variance data><CR><LF>"
=> 4 + 8 + 2 + (1+NumMeas+NumMeas)x8 + 2

FilterID - NumStates => size
1 - 4 => 4 + 8 + 2 + (1+4+4)x8 + 2 = 88
2 - 10 => 4 + 8 + 2 + (1+10+10)x8 + 2 = 184
3 - 10 => 4 + 8 + 2 + (1+10+10)x8 + 2 = 184
4 - 6 => 4 + 8 + 2 + (1+6+6)x8 + 2 = 120
5 - 6 => 4 + 8 + 2 + (1+6+6)x8 + 2 = 120
6 - 8 => 4 + 8 + 2 + (1+8+8)x8 + 2 = 152
7 - 8 => 4 + 8 + 2 + (1+8+8)x8 + 2 = 152
8 - 8 => 4 + 8 + 2 + (1+8+8)x8 + 2 = 152
9 - 8 => 4 + 8 + 2 + (1+8+8)x8 + 2 = 152
10 - 4 => 4 + 8 + 2 + (1+4+4)x8 + 2 = 88
11 - 4 => 4 + 8 + 2 + (1+4+4)x8 + 2 = 88
12 - 4 => 4 + 8 + 2 + (1+4+4)x8 + 2 = 88
13 - 4 => 4 + 8 + 2 + (1+4+4)x8 + 2 = 88
14 - 6 => 4 + 8 + 2 + (1+6+6)x8 + 2 = 120
15 - 6 => 4 + 8 + 2 + (1+6+6)x8 + 2 = 120

=> 1896 bytes

=> 8512 bytes per 1.28 sec

Sub-Total: 682.56 MByte/Day

OVERALL TOTAL MAX DATA TO BE LOGGED IN ADM: 734 MByte/Day

APPENDIX F DIINS XDM SOFTWARE OPERATION

F.1 Introduction

This appendix contains a brief description of the manner in which the software within the DIINS XDM operates. The information contained in this appendix has been generated using the Rational Rose CASE tool. **Figure F-1** presents the Use Case view of DIINS for the two cases considered, those of "Initialization" and "Operational Mode". The DIINS ADM will require the inclusion of another Use Case, namely that of maintenance, which will be accomplished in part via the enhanced operator interface to be provided to the ADM.

F.2 DIINS XDM Use Cases

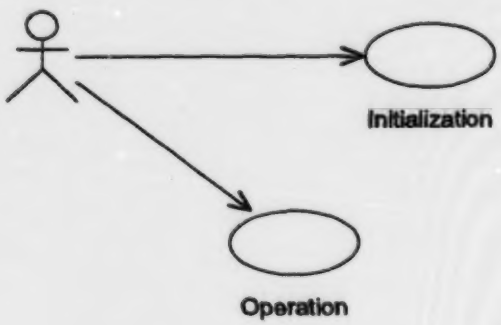

Figure F-1 DIINS XDM Use Case Model

F.3 Initialization

Figure F-2 illustrates the timing involved in the various tasks employed in the DIINS XDM. In the Sequence Diagram presented, the various software objects are indicated across the top of the diagram, and time advances down the page. In a Sequence Diagram the arrows between the various vertical bars represent calls to the "methods" or operations associated with the various objects. These arrows therefore represent procedure, function or task entry calls in the DIINS Ada software. Information may flow (not shown in all cases) in the direction of the arrow, in the opposite direction to the arrow or in both directions, and represents the parameters passed during a call and the results returned, if any. In some cases the arrow has been labeled as "with", and this is just to indicate that there is no actual method call involved, but rather a triggering "with" relationship exists between the two objects at either end of the arrow. This indicates that the object at the sharp end of the arrow is "withed" in the Ada sense by the other object, causing the object to elaborate or come into existence and start performing its task. The diagrams have been annotated with notes in order to make them self explanatory when read from the top to the bottom of the figure.

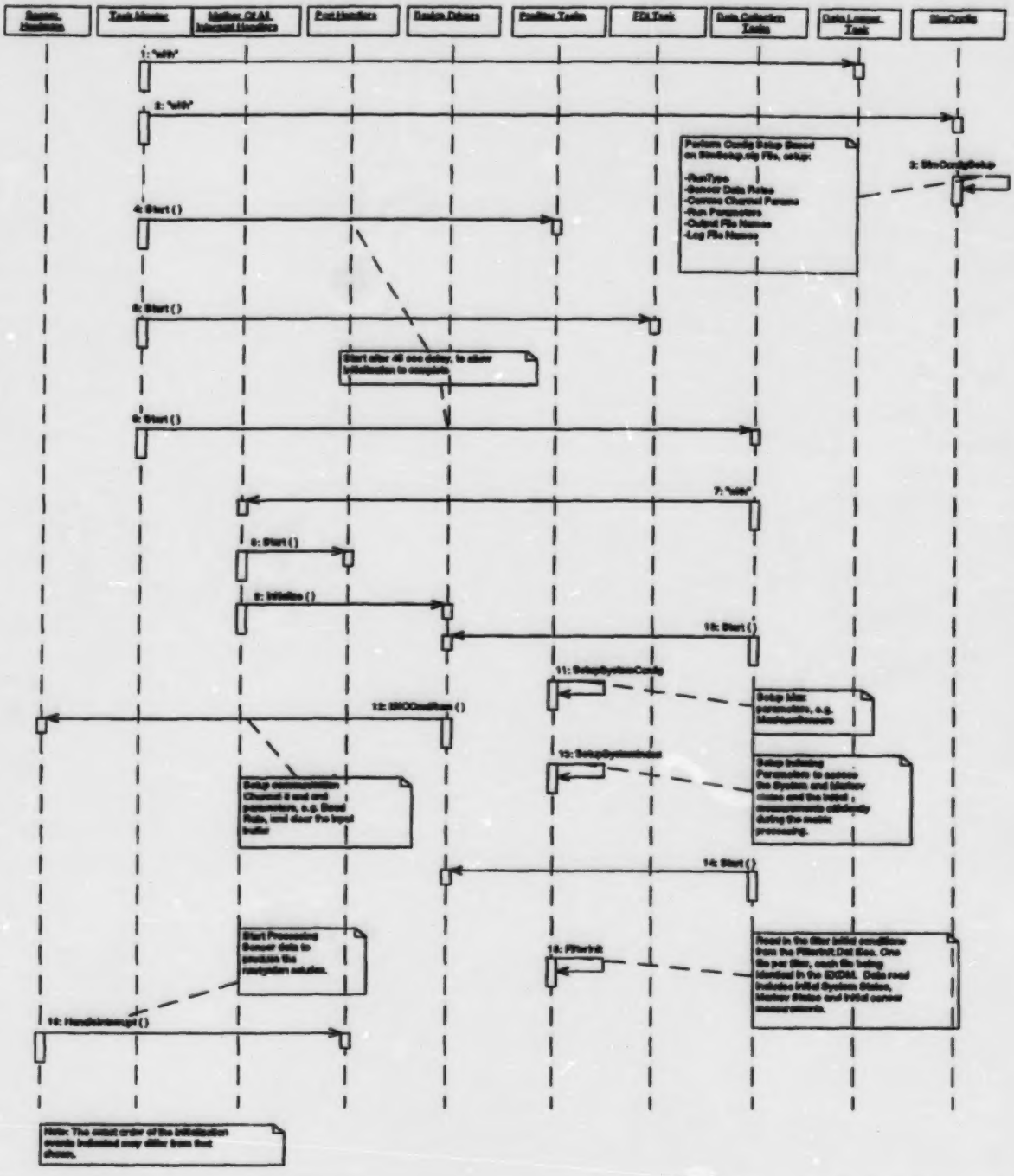

Figure F-2 DIINS XDM Software Initialization

F.4 Operational Mode

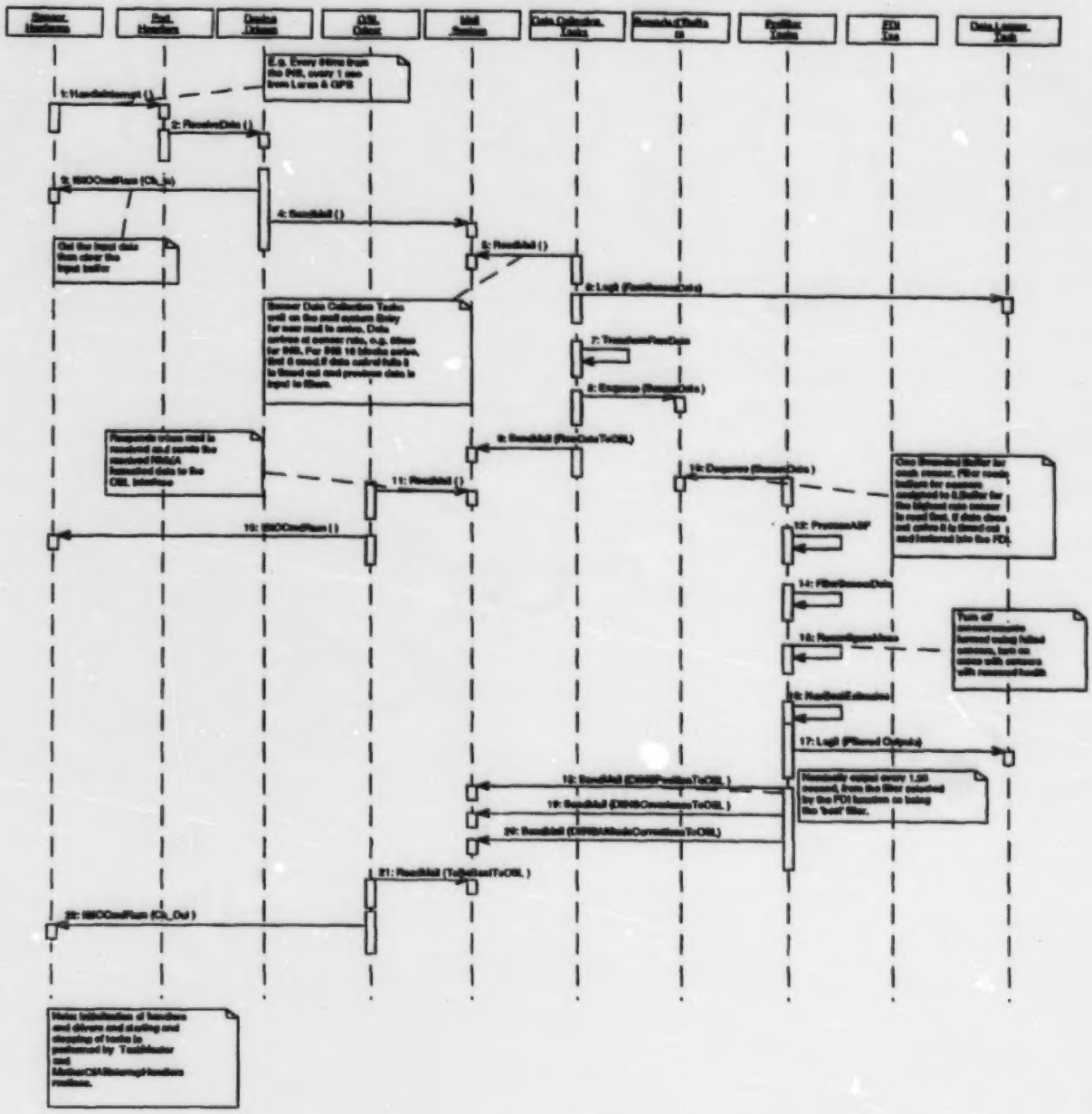

Figure F-3 DIINS XDM Software Operational Mode

APPENDIX G DIINS ADM SOFTWARE OPERATION

G.1 Introduction

This appendix presents sequence diagrams of the ADM operation generated using the Rational Rose CASE tool. These diagrams represent the software operations that take place between the significant Computer Software Components (CSCs) in the ADM.

In order to produce the model contained in this appendix, certain decisions have been made as to how the sensor specific configuration data within the ADM will be organized. The issues involved and the proposed solution are presented in Appendix I.

The ADM model presented in this section took as a starting point the XDM software operation model defined in Appendix F, and Figures G-1 and G-4 follow the XDM model closely. This initial model has been extended to include, in particular, the new HCI features to be provided in the ADM to enable the user (system operator or maintainer) to more readily configure and monitor the operation of DIINS when in port or at sea. The behavioral description defined in the sequence diagrams presented in this appendix should be read in conjunction with the relevant HCI interface forms defined in Appendix D, and the SHINADS to DIINS data transfer NMEA formatted messages defined in Appendix H. The content of these three appendices (D, G and H) fully define the SHINADS to DIINS ADM interface and system operational behavior to the level required in this systems analysis and design report.

The diagrams presented in this appendix have been simplified to some extent. They do not show the details of all of the interactions that take place. In particular, the NMEA encoder and decoder objects are not shown explicitly, but the use of these objects is implied when two configuration items that communicate via NMEA format are shown interacting on the diagrams, e.g. the HCI receiving information from the Data Collector or sending information to the SystemConfigController. In addition, the details of MotherOfAllInterruptHandlers and the PortHandlers have been omitted from the ADM diagrams in order to leave more room for other objects. The behavior of these objects is the same as in the XDM case.

There are two main points of contact between the DIINS HCI and the DIINS Ada software. These points are the Data Collection SWCI (DIINS->HCI) and the SystemConfigController (HCI->DIINS). The transfer of data between the HCI and these two SWCIs is via the NMEA format defined in Appendix H.

G.2 DIINS ADM Use Cases

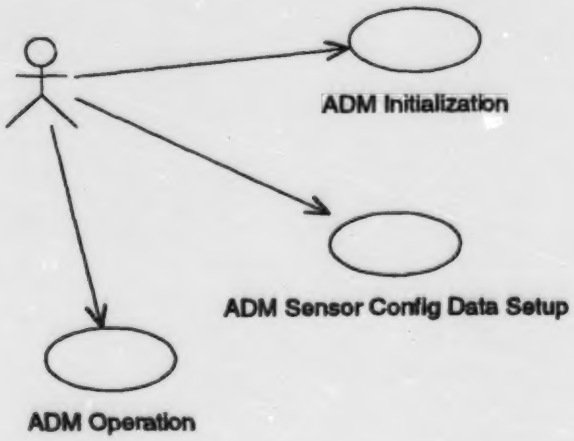

Figure G-1 DIINS ADM Use Case Model

G.3 DIINS ADM Initialization

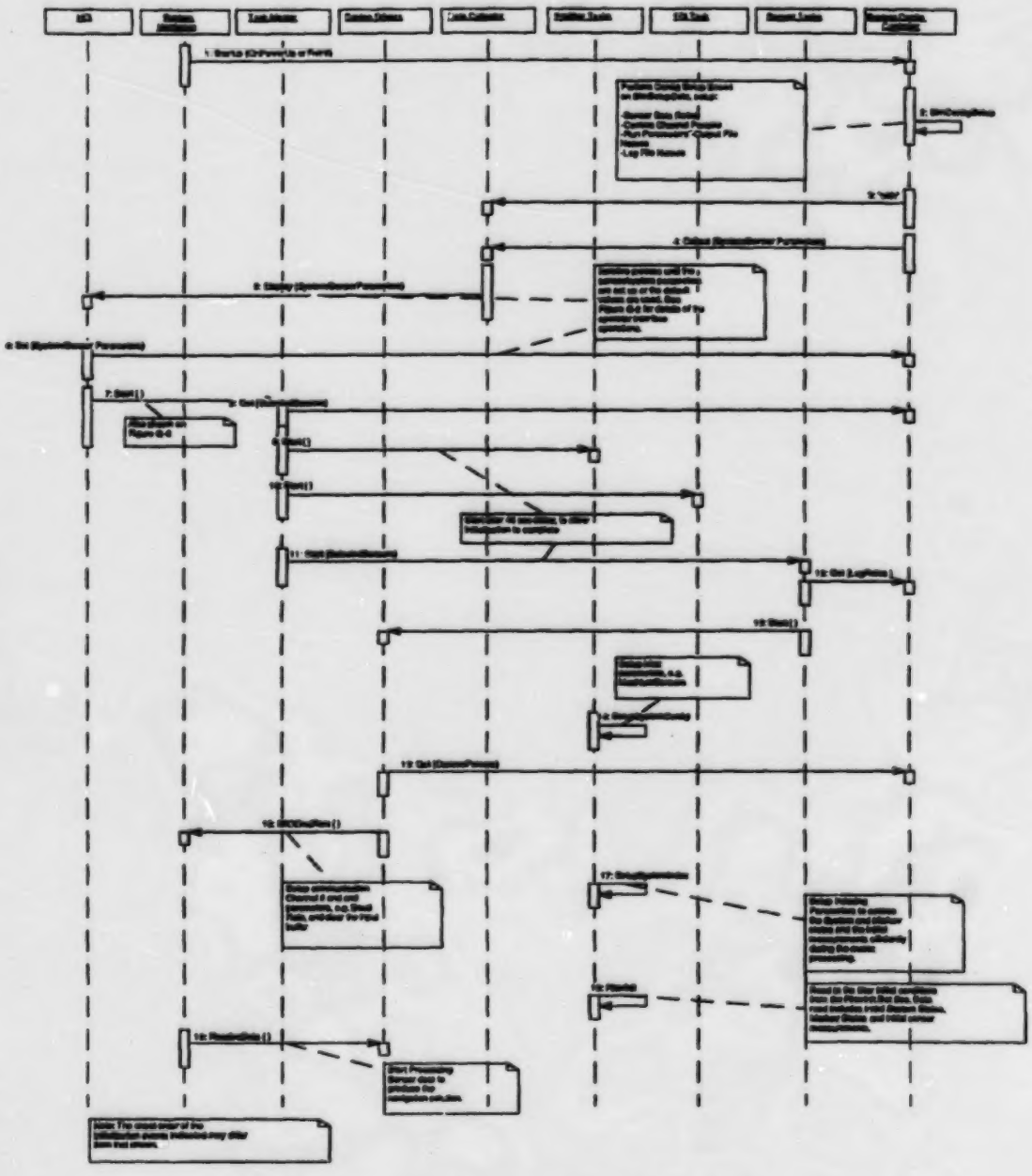

Figure G-2 DIINS ADM Initialization

G.4 DIINS ADM Sensor Configuration Data Setup

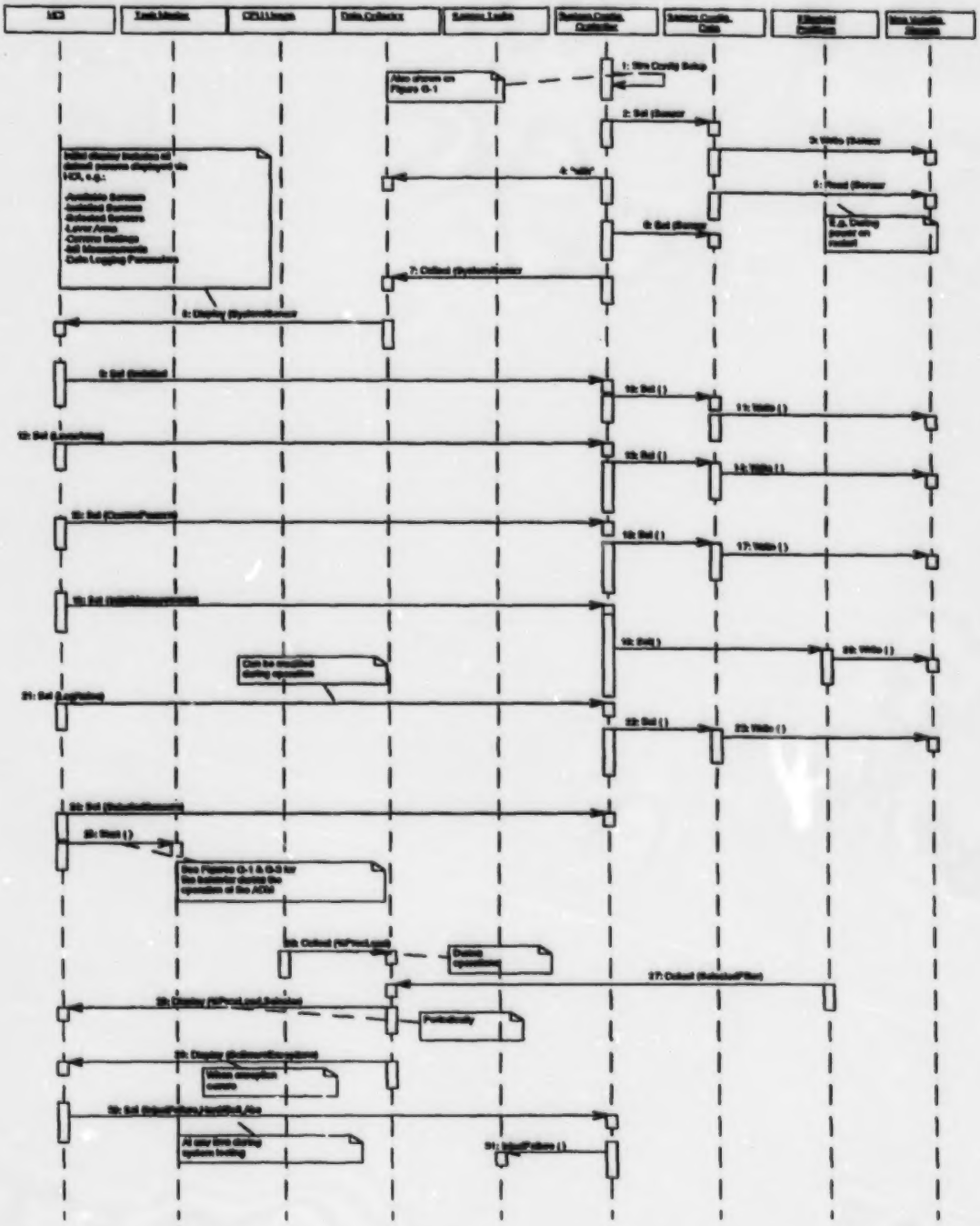

Figure G-3 DIINS ADM Sensor Configuration Data Setup

G.5 DIINS ADM Operation

In Figure G-4 the OSL Drivers are shown; whereas, the new Data Collection SWCI is also involved in the transfer of the data, see Appendix C for details.

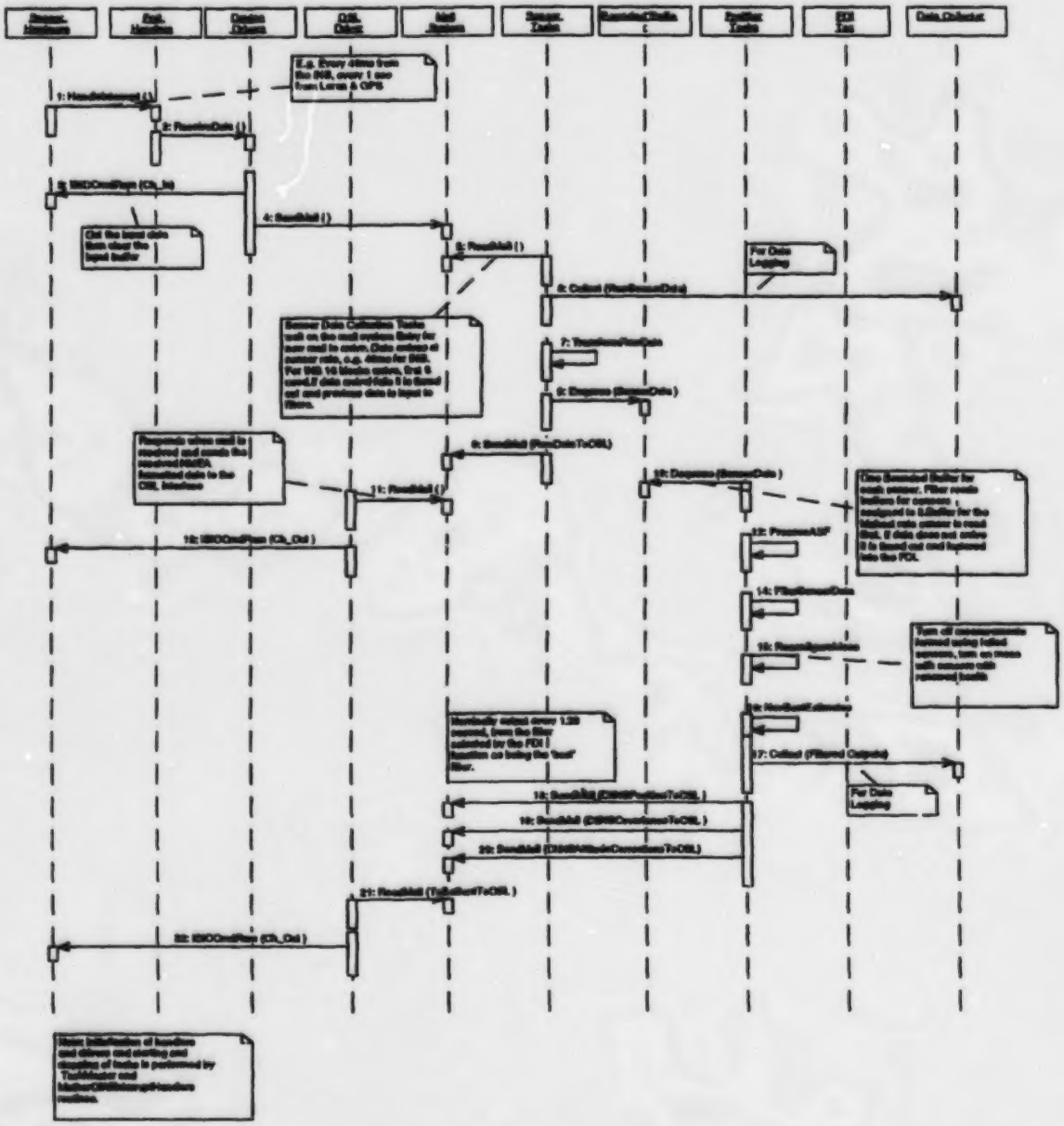

Figure G-4 DIINS ADM Operation

APPENDIX H NEW SHINNADS TO DIINS INTERFACE REQUIREMENTS

H.1 Introduction

This appendix defines the new NMEA sentences required in order to produce the DIINS ADM from the XDM. These new sentences are required in order to implement the new HCI requirements for the ADM. The new sentences are used to communicate information between the HCI software and DIINS.

The sentence definitions presented in this appendix are preliminary, and will be subject to verification against the NMEA sentence formats defined in Reference[8]. In addition, the sentence formats may be modified during the detailed design in order to optimize the data transfer format. For example, rather than implement a form based sentence scheme as defined in this appendix, it may be more appropriate to partition the data in another manner. For example, all of the data associated with a particular sensor could be sent in a single sentence, e.g. its identifier, type, lever arm etc. The pros and cons of the alternative schemes will be examined as part of the software detailed design.

The following format definitions apply in the custom sentence definitions given in this section:

DRO Manufacturers mnemonic code for Defence Research Ottawa
*hh Sentence checkcode
<CR> Carriage Return
<LF> Linefeed

H.2 Sensor Available For Installation (Direction: DIINS->SHINNADS)

$PDRO3,tttt,ssss,*hh<CR><LF>

tttt = Sensor Type

Valid sensor types are as follows:

- INS1
- INS2
- GPS
- dGPS
- Loran

ssss = Sensor
Valid sensors are as follows:

- MK29 (INS)
- LTN90 (INS)
- MK49 (SINS) Note: Applicable to INS type.
- AshtechXII (GPS)
- TrimbleTANSPY (GPS)
- TrimbleVector (GPS)
- UofC (dGPS)
- LeicaMX94XX (dGPS)
- InternavLC360 (Loran)

H.3 Installed Sensor Model (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

$PDRO4,tttt,ssss,*hh<CR><LF>

tttt and ssss = same as for PDRO3.

H.4 Selected Sensor (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

$PDRO5,tttt,ssss,*hh<CR><LF>

tttt and ssss = same as for PDR03.

H.5 Sensor Lever Arm (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

$PDRO6,tttt,ssss,xxxx.xx,yyyy.yy,zzzz.zz,*hh<CR><LF>

Where:

- tttt and ssss = same as for PDR03.
- x, y and z are the sensor lever arm coordinates in metres.

H.6 Sensor Communications Setup (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

$PDRO7,tttt,ssss,ChID,BaudRate,NumBits,Parity,Stop,Handshake,*hh<CR><LF>

Where:

ChID: In range 00 - 99

BaudRate: {2400|9600|19200}

NumBits: {7|8}

Parity: {None|O|E} i.e. Odd or Even

Stop: {1|2}

Handshake: {None|TBD}

H.7 Initial Measurements (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

$PDRO8,MeasID,Sigma,ResidTest,FailFrac,FailTime,PassFrac,PassTime,*hh<CR><LF>

where:

- MeasID: alpha-numeric sensor measurement ID , e.g. X_GPS2_POS_MEAS
- Sigma: f.f
- ResidTest: f.f
- FailFrac: f.f
- FailTime: f.f
- PassFrac: f.f
- PassTime: f.f

where "f.f" = floating point number

H.8 Data Logging

H.8.1 Selected Sensors For Logging (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

$PDRO9,s1,s2,s3,s4,s5,*hh<CR><LF>

where:

s1 = INS1
s2 = INS2
s3 = GPS
s4 = dGPS
s5 = Loran

si = 0 : Not selected

si = 1: Selected

H.8.2 Selected Filters For Logging (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

$PDRO10,s--s,*hh<CR><LF>

where:

s--s: indicates the selection status of each filter; s = 0 Not Selected, s = 1 Selected

Note: The order of the filters in s--s is Filter 1 to Filter n, e.g. n = 7 or n = 15.

H.8.3 Logging Rate Scales (Direction: SHINNADS->DIINS, DIINS->SHINNADS)

$PDRO11,r0,r1,r2,r3,r4,r5,*hh<CR><LF>

where:

r0: Overall Rate Scale, format = f.f in range 0<= r<=1

r1 = INS1
r2 = INS2
r3 = GPS
r4 = dGPS
r5 = Loran

ri: Sensor Rate Scale, format = f.f in range 0<= r<=1

H.9 Failure Injection (Direction: SHINNADS->DIINS)

$PDRO12,FailureMode,AbsRel,FailTime,Duration,*hh<CR><LF>

where:

FailureMode:	H = Hard S = Soft
AbsRel:	A = Absolute timing R = Relative timing
FailTime:	hhmmss.ss (absolute or relative)

Duration: hhmmss.ss, failure injection duration

H.10 System Status

H.10.1 Processor and Filter Status (Direction: DIINS->SHINNADS)

$PDRO13,ProcLoad,SelFilter,*hh<CR><LF>

where:

ProcLoad: f.f

SelFilter integer number, in range 1 to number of filters.

H.10.2 Software Events (Direction: DIINS->SHINNADS)

$PDRO14,t,EventText,*hh<CR><LF>

where:

t: hhmmss.ss, time of occurrence of the event

EventText: character expression, with a maximum number of 40 characters, identifying the Software Exception, FDI Log or Filter Log events that occur.

APPENDIX I DIINS ADM SENSOR CONFIGURATION DATA ORGANIZATION

I.1 Introduction

In the XDM the configuration data is contained in three main file types, see Appendix B for details of the file content:

- SimSetup.Cfg (1x off)
- FilterInitx.Dat (n x off, where n is the number of filters, each file is identical in the XDM)
- Filter.Cfg (n x off)

One issue considered during the systems analysis has been whether the sensor data should be reorganized in order to group the information for a particular sensor together (c.f. object-oriented) rather than the current approach, whereby all of the data for the various sensors is grouped together.

I.2 Configuration Data Organization

In the case of the **SimSetup.Cfg** file, the main sensor specific information consists of the sensor ID, sensor update rate, communications channel ID and the channel parameter settings for the sensor. All of the other data in the file is independent of the particular sensors used, and can be derived from the sensors selected for use. In the case of the number of FilterInit files defined, the possibility of reducing the FilterInit files to a single file will be investigated. Such decisions will not be made at this stage, as the full ramifications of such modifications need to be understood, and hence are best left to the detailed design and coding phases.

In the **FilterInitx.Dat** files, the only data associated with specific sensors is the sensor lever arms.

In the case of the **Filter.Cfg** files, which define the particular sensor configuration seen by a particular filter, the content of the configuration files that is specific to the particular sensor selected is the MeasModel, MeasCorrelation and NoMeasTimeTrigger. These parameters are not ones that are required to be modified in the operational environment, but will have to be defined accordingly when a new sensor is added.

One of the main reasons to consider grouping the data for each sensor separately would be in order to make it easier to add a new sensor, i.e. adding the data necessary to define the sensor in a single place in the software. Bearing in mind that new sensors would only be added to DIINS in the development environment (due to the complexity of the task) the sensor information can be hardcoded into the code, e.g. default sensor parameter values can be defined in the development environment, some of which may be modified by the system user during DIINS

operation. To follow good software practices this hard coding should be done in a manner so that data can be changed without having to recompile the majority of the code associated with DIINS, just recompile the single module affected. Some of the information currently manually updated in the DIINS configuration files prior to a DIINS run should ideally be derived from the current selected sensor configuration.

From the above it can be seen that the following information is necessary to define a new sensor to be used in DIINS:

- Sensor ID (e.g. SPERRY_MK29);
- Sensor Type (e.g. INS, as distinct from the sensors role, e.g. as INS2);
- Sensor Update Rate (e.g. 1.28 sec);
- Sensor Comms Channel ID and Parameters (e.g. Baud Rate);
- Lever Arm;
- MeasModel (defined in development lab);
- Measurement Correlation (defined in development lab); and
- No Measurement Trigger Time (defined in development lab).

With reference to the proposed system design in Appendix C, the obvious place to group sensor specific data appears to be the Real_Sensor object defined in Figure C-1. As seen in Appendix C, it is currently planned to make the operational real sensors instantiations of a generic Real_Sensor package. This package is currently shown as having three generic parameters, namely the type Raw_Sensor_Data_Type, the Sensor_Identifier string, and a procedure Transform_Raw_Record to transform the raw sensor data into the format used within DIINS.

The parameters indicated above, e.g. LeverArm, could be added to the list of parameters passed to the Real_Sensor when it is instantiated. In fact, all of the required sensor initialization data could be passed to the generic package as a record structure if desired. One problem that arises is that a dialog has to take place with the user during system initialization in order to define the sensors installed and sensors selected for operation. This dialog includes defining the sensor lever arms and the communications parameters. This information needs to be obtained before the Real_Sensors are instantiated, so that the information can be passed to the sensor during instantiation. Indeed, some of the information has to be obtained for installed sensors that may never be selected for operation, hence would not have a task associated with them, which might otherwise be involved in maintaining the sensor data. The above implies that the sensor configuration data for the ADM should be stored independent of whatever "real sensor" tasks are instantiated.

In the XDM, the above configuration information is contained in the configuration files. In the ADM, the configuration file information will have to be placed in writable non-volatile memory, so that, for example, the last system configuration can be maintained through a power dropout. The non-volatile information has to maintain the data structures for the Available, Installed and Selected sensors as defined by the system user. As the information has to be held independent of Real_Sensor task instantiation (e.g. for installed but not selected tasks) there does not appear to be a strong argument to place the sensor configuration information in the Real_Sensor tasks.

I.3 ADM Cofiguration Data Solution

The solution adopted for the ADM to the problem of being able to easily add in a new sensor for the ADM will be to place all of the information for the sensor (indeed for all of the sensors) into a single configuration data structure (SimSetup). Currently there may be areas of code which have to be changed when a new sensor is added to the configuration files. If there are such changes required in the XDM code, an attempt will be made to eliminate them during the ADM detailed design and coding. The design goal is to have all of the changes necessary to introduce a new sensor confined to a single data block in SimSetupData, as indicated in the following example proposed data format:

```
_*************************************************************************
----------[ Content-type: text/plain; name=SimSetupData ]----------

--This configuration data contains the definition of the sensors currently handled
--(Available Sensors) by the DIINS ADM. Separate data entry sections have been
--provided for both the primary and secondary sensors, as the default parameters
--contained in the data may differ between the two sensors of the same type. The default
--parameters will be used to initiate the data structures (e.g. list/array of data records) for
--the Available, Installed and Selected sensors, which will be used to interface with the
--user via the ADM HCI. Any modification via user inputs of the default values will be
--performed on these run-time data structures, will be re-displayed to the user and will be
--saved in non-volatile storage.

_*************************************************************************
-- SPERRY MK29 INS1
_*************************************************************************

Sensor Model:          SPERRY_MK29_INS1

DataDeltaT (sec):      1.28

Comms Port:            1
Baud Rate:             BAUD_9600
No. Bits:              BITS_8
Parity:                PARITY_NONE
Stop Bits:             STOP_1
Handshake:             HANDSHAKE_NONE

Lever Arm (X,Y,Z):     0.0      0.0     0.0     metres

Meas Model:            LINEAR
Meas Correlation:      UNCORRELATED
```

```
NoMeasTimeTrigger: 1.0 MINUTES

Data Log Rate (sec):  1.28

--*****************************************************************************
-- Other Sensors TBD
--*****************************************************************************

TBD


--*****************************************************************************
```

I.4 Configuration Data Ada Package Structure

The following illustrates the kind of Ada package structure that might be appropriate to define the SensorConfigDataType, and to set and retrieve components of objects of that type to form the higher level data structures of the list/array of Available, Installed and Selected sensors planned to be used during the operation of the ADM.

```
--*****************************************************************************
--*****************************************************************************
-- PACKAGE SPEC SensorConfigData
-*****************************************************************************
--*****************************************************************************

package SensorConfigData is

        type SensorConfigDataType is private

--*****************************************************************************
-- CONSTRUCTORS
-*****************************************************************************

        procedure Create (SensorCfgDataObject:...);

        procedure Set (TheSensorModel:...;InSensorCfgDataObject: ...);

        procedure Set (TheDataDeltaT:...;InSensorCfgDataObject: ...);

        procedure Set (TheCommsParams:...;InSensorCfgDataObject: ...);

        procedure Set (TheLeverArm:...;InSensorCfgDataObject: ...);
```

```
        procedure Set (TheMeasModel:...;InSensorCfgDataObject: ...);

        procedure Set (TheLogRate:...;InSensorCfgDataObject: ...);

--**************************************************************************
-- SELECTORS
-**************************************************************************

        function GetSensorModel (FromSensorCfgDataObject: ...) return ...;

        function GetDataDeltaT(FromSensorCfgDataObject: ...) return...;

        function GetCommsParams (FromSensorCfgDataObject: ...) return...;

        function GetLeverArm(FromSensorCfgDataObject: ...) return...;

        function GetMeasModel (FromSensorCfgDataObject: ...) return...;

        function GetLogRate (FromSensorCfgDataObject: ...) return...;

private

        type SensorConfigDataType is
                SensorModel: ...;
                DataDeltaT:...;
                CommsParams:...;
                LeverArm:...;
                MeasModel:...;
                LogRate:...;
        record

end record;

end SensorConfigData;

--**************************************************************************
--**************************************************************************
-- PACKAGE BODY SensorConfigData
-**************************************************************************
--**************************************************************************

package body SensorConfigData is
.
.
.
```

```
end SensorConfigData;

--*****************************************************************************
```

Note: The types not explicitly shown in the above package example, and indicated as "..." could be other user defined types, e.g.:

```
type CommsParamType is
    record
    .
    .
    .
    end record;
```

UNCLASSIFIED
SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Prior Data Sciences Ltd.
240 Michael Cowpland Drive
Kanata ON K2M 1P6

2. SECURITY CLASSIFICATION (overall security classification of the document, including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

DIINS ADM Phase I Report: System Design Review (U)

4. AUTHORS (Last name, first name, middle initial)

Jackson, G., Davis, T.

5. DATE OF PUBLICATION (month and year of publication of document)	6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)	6b. NO. OF REFS (total cited in document)
30 Jan 1998	135	9

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

CONTRACTOR REPORT

SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

Defence Research Establishment Ottawa
Department of National Defence
Ottawa ON K1A 0Z4

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written)
Project 1bd17	W7714-6-0043

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)
97-0119-K DRDC CR 1999-117	

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

UNLIMITED

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

This report records the results of the studies conducted to review the system requirements and design during Phase I of the development of the Dual Inertial Integrated Navigation System (DIINS) Advanced Development Model (ADM). The requirement addressed by the studies documented in this report was to investigate the system requirements and design for the ADM based on the resulting system design and experience gained during the experimental model (XDM) development. The results of the analysis reported in this document will be used to drive the developments in the subsequent phases of the ADM..

#512193

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

DIINS
Navigation Systems
Navigation Software
Software Design